RESUMEN

"IMPLEMENTACIÓN DE ALGORITMOS PARA EL CÁLCULO DE FLUJO ÓPTICO EN UNA SECUENCIA DE IMÁGENES"

En base a la teoría de flujo óptico, en esta tesis se implementan algoritmos por software para calcular el movimiento a partir de una secuencia de imágenes. El objetivo de este trabajo es crear un sistema para el cálculo de flujo óptico que facilite un desempeño en tiempo real bajo una plataforma de fácil acceso. La plataforma consiste en utilizar un lenguaje o herramienta de programación de uso común en la implementación de los algoritmos para el cálculo de flujo óptico. Se emplea una cámara web como sensor de video y una computadora de propósito general para el procesamiento de imágenes. El desarrollo del sistema para cálculo de flujo óptico se divide en tres etapas: en la primera etapa se desarrollan programas prueba utilizando la herramienta de procesamiento MATLAB. Se comprueba la detección de flujo óptico y se valora el tiempo de procesamiento de los algoritmos. En la segunda etapa se diseña un modelo a bloques de un sistema de detección de flujo óptico en tiempo real. Aquí, se utiliza la herramienta *"Video and Image Processing Block Set"* del software Simulink que se integra a MATLAB. El modelo se basa en fases establecidas para el cálculo de flujo óptico en secuencias de imágenes. En esta etapa se valora tiempo de procesamiento, sensibilidad al ruido, y la detección de vectores de flujo óptico válidos. En la tercera etapa se desarrolla el sistema para el cálculo de flujo óptico utilizando lenguaje C y la biblioteca para procesamiento de imágenes OpenCV . El sistema se basa en el modelo que se establece en la segunda etapa con un cambio en las fases para la detección de flujo óptico en secuencias de imágenes. Se obtiene como resultado, un sistema para calcular el flujo óptico en secuencias de imágenes que proporciona valores de magnitud y dirección del movimiento. Estos datos se utilizan para alimentación a otros módulos de procesamiento de mayor nivel, tales como sistemas de navegación y seguidores de objetos. Se realizan aplicaciones del sistema de cálculo de flujo óptico para comprobar su funcionalidad. Una de estas aplicaciones es el reconocimiento de patrones y su seguimiento, donde la detección de flujo óptico se enfoca sólo al área determinada. Otra aplicación es el enfoque de un microscopio para análisis cuantitativo del movimiento de microestructuras. En ambos casos se utilizan las funciones implementadas de detección de movimiento por flujo óptico, obteniendo resultados satisfactorios.

ABSTRACT

"IMPLEMENTATION OF ALGORITHMS FOR THE OPTICAL FLOW COMPUTATION OF AN IMAGE SEQUENCE"

Based on the optical flow theory, in this thesis software algorithms are implemented to compute the motion present on an image sequence. The objective of this work is to create a system for optical flow computation that supports performance in real time with a readily accessible platform. The platform consists of using a language and tool for common programming of algorithms applied to the optical flow implementation. A Web camera is used as a video sensor and a general-purpose computer for the image processing. The development of the optical flow computation system is divided in three stages: in the first stage, test programs are developed using the processing software tool MATLAB. The optical flow detection is verified and the algorithms processing time is valued. In the second stage a block model of real time optical flow detection system is designed. Here, the *"Video and Image Processing Block Set"* tool of the Simulink software in MATLAB is used. The model is based on the phases established for optical flow calculation of an image sequence. In this stage, processing time, noise sensitivity, and valid optical flow vectors are valued. In the third stage, the optical flow computation system is developed using C language and the OpenCV library for image processing. The system is based on the model produced in the second stage with a change in the phases for the optical flow detection. As a result, a system is obtained to calculate the optical flow in an image sequence that provides values of magnitude and direction of movement. These data are used for feeding to other higher level processing modules, such as positioning systems and objects tracking. Applications of the optical flow computation system are employed to verify their functionality. One of these applications is the pattern recognition and its tracking, where the optical flow detection focuses only to the detected area. Another application is the quantitative analysis of microstructures movement. In both cases the implemented functions of movement detection by optical flow are used, obtaining satisfactory results.

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE INGENIERÍA

“IMPLEMENTACIÓN DE ALGORITMOS PARA EL CÁLCULO DE FLUJO ÓPTICO EN UNA SECUENCIA DE IMÁGENES”

TESIS

QUE PARA OBTENER EL GRADO DE
Maestro en Ingeniería

PRESENTA
Mónica Valenzuela Delgado

Director
Dr. Miguel Enrique Bravo Zanoguera

Mexicali, Baja California

AGRADECIMIENTOS

A Dios

Por permitirme estar aquí y continuar con mis metas.

A mis sinodales

Por su apoyo, sus consejos y recomendaciones, pero sobre todo, por la paciencia que han tenido conmigo.

A mi director de tesis Miguel Enrique Bravo Zanoguera

Por su confianza y comprender mi forma de trabajar.

A mi alumno José Carlos Guerrero Buenrostro

Por su colaboración en el proyecto.

Al M.C. Juan Guillermo Anguiano Silva

Por su apoyo incondicional para la culminación de esta meta.

A mi familia y a todos los que me alentaron y que de algún modo contribuyeron a la culminación de esta fase profesional de mi vida.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO I

INTRODUCCIÓN

1.1 Motivación

De la gran variedad de temas a explorar en cuanto a visión artificial tales como identificar formas, colores, restaurar imágenes entre otras, la detección de movimiento en una forma visual, es una de las capacidades más útiles tanto de la visión humana como de animales e incluso indispensable para ciertos insectos. Por consiguiente, la medición de movimiento en el entorno que nos afecta, es parte esencial de las actividades diarias en una sociedad moderna, como lo son observaciones meteorológicas, sistemas de vigilancia y recientemente sistemas de navegación controlados por visión. Es por ello que la visión dinámica, es un tema de sumo interés entre los investigadores. Desde el punto de vista práctico, según Ballard y Brown (1982) y Pajares y De la Cruz (2002) los problemas relacionados con el movimiento se agrupan básicamente en: detección de movimiento, detección y localización de objetos en movimiento y detección de propiedades 3D del objeto a partir de vistas 2D del objeto en movimiento.

a) **Detección de movimiento**. Se trata de registrar cualquier movimiento; es útil en el campo de la seguridad y se emplea una cámara estática.

b) **Detección y localización de objetos en movimiento**. Se utiliza una cámara estática y los objetos se mueven en la escena, o la cámara se mueve mientras los objetos son estáticos, o ambas cosas a la vez. Ejemplos de aplicaciones en este campo son las observaciones meteorológicas y predicción del movimiento de las imágenes tomadas por satélite, así como la predicción del tráfico de vehículos en una ciudad.

c) **Detección de propiedades 3D del objeto a partir de vistas 2D del objeto en movimiento.** Algunas aplicaciones en este campo son la identificación de piezas

industriales o mecánicas, la automatización de procesos y la identificación de objetos para la navegación en robótica.

El punto de interés de este trabajo es la medición de movimiento, sin considerar la identificación de objetos: su localización o determinación de sus propiedades. Existen una gama de aplicaciones donde se requiere conocer solamente el comportamiento dinámico del campo de visión, es decir, determinar la magnitud del movimiento en la escena y su orientación; por ejemplo, en imágenes meteorológicas, imágenes de fluidos y sistemas de navegación por visión.

1.2 Planteamiento del problema

Las técnicas de visión extraen información para que un sistema lleve a cabo determinadas acciones. Las acciones que se realicen con esta información, dependen de la aplicación en particular. Para el caso específico del cálculo de flujo óptico, la problemática de algunas técnicas de visión es que la mayoría han sido implementadas en arquitecturas hardware (Zuloaga, Martín y Ezquerra, 1998; Cobos, 2001), las cuales permiten una mayor velocidad en el procesamiento de datos en tiempo real, pero complicadas de implementar y con el inconveniente de que son diseñadas para sistemas específicos de visión.

En algunas investigaciones se han implementado algoritmos por software utilizando una computadora de propósito general para la determinación de movimiento en secuencias de imágenes. Estas secuencias provienen de archivos almacenados y no necesariamente de procesamiento en tiempo real (Horn y Schunck, 1981 Lucas y Kanade, 1981; Baker, Gross, Matthew e Ishikawa, 2003). En otros proyectos se utilizan solo dos cuadros de escena para comprobar con que exactitud, los métodos empleados son capaces de determinar el campo del movimiento (Kusina, 2000).

Considerando el estado del arte, no existe un desarrollo de software, para el cálculo de flujo óptico en tiempo real, que se haya aceptado como una plataforma confiable y de fácil implementación. Por consiguiente, la problemática de esta investigación es la siguiente: ¿Es posible implementar por software algoritmos para el cálculo de flujo óptico

a partir de escenas dinámicas procesadas en tiempo real? Para tal propósito se requiere una computadora personal y un sensor de video de propósito general, de tal modo que se obtenga como resultado parámetros de movimiento (magnitud y dirección) aplicables a sistemas de visión, partiendo de los algoritmos establecidos para cálculo de flujo óptico.

1.3 Antecedentes

Existen trabajos de investigación cuyo objetivo es el análisis de movimiento en una secuencia de imágenes. Algunos de los trabajos consultados implementan algoritmos conocidos mediante software, donde analizan escenas provenientes de archivos o fuera de línea, es decir, no son en tiempo real y en ocasiones utilizan imágenes sintéticas para el análisis (Horn y Schunck, 1981; Lucas y Kanade, 1981; Kusina, 2000; Baker *et al.* 2003). Otras investigaciones realizan la detección del movimiento en tiempo real, pero suele ser por medio de hardware y regularmente en aplicaciones específicas (Zuloaga *et al.* 1998; Cobos, 2001). A continuación se describen los antecedentes que contribuyen con su información al desarrollo de esta tesis.

Berthold K.P. Horn y Brian G. Schunck del Instituto de Tecnología de Massachusetts, E.U.A., formularon en 1981 una de las ecuaciones más importantes que se utilizan en el procesamiento de secuencias de imágenes conocida como, Ecuación Restringida del Flujo Óptico *(Optical Flow Constraint Equation o Spatio-Temporal Constraint Equation)*. En su artículo suponen que se tiene una secuencia de imágenes donde se cumplen las siguientes condiciones:

- La iluminación de los objetos presentes en las imágenes es uniforme, la reflexión de los objetos varía con suavidad y sin discontinuidades espaciales.
- No existen objetos que se oculten unos a otros.
- No hay nuevos objetos en la secuencia de imágenes.

Según las condiciones anteriores, se afirma que la intensidad de un punto proyectado en la imagen no cambia, lo que cambia es su posición. Una implementación iterativa se demuestra en este artículo utilizando imágenes de baja resolución y trabajando

con secuencias sintéticas de la imagen, de tal modo que comparan los resultados de la implementación con los valores exactos calculados utilizando las ecuaciones correspondientes al método matemático. En la implementación realizada por Horn y Schunck (1981) se inducen algunos problemas comunes en el análisis de imágenes como la adición de ruido, cálculo de gradientes en superficies sin textura y determinación de movimiento en objetos uniformes como el caso de una esfera girando, de donde no es posible calcular el flujo óptico debido a que no se presenta ningún cambio entre los cuadros de imagen de la secuencia. Los resultados obtenidos por Horn y Schunck en su investigación son: la reducción de errores en los cálculos y la propuesta de soluciones iterativas a los problemas del análisis de movimiento, mencionados anteriormente. Debido a los resultados obtenidos por estos autores, muchas otras investigaciones referentes al tema de la detección de movimiento por flujo óptico, han tomado como base la Ecuación Restringida de Flujo Óptico para desarrollar algoritmos de detección de movimiento. En este trabajo de tesis se han incluido métodos que utilizan esta ecuación como base y que además emplean otras técnicas para minimizar los efectos producidos por las restricciones que demanda la ecuación.

Otro de los artículos revisados es el de los autores Bruce D. Lucas y Takeo Kanade (1981). Estos investigadores presentan una técnica de registro para el análisis imágenes aplicables a la solución de problemas de visión, tales como visión estereoscópica, reconocimiento de imágenes y análisis de movimiento entre otras. Esta técnica utiliza los gradientes espaciales de las imágenes para la búsqueda de posiciones entre una imagen y otra. Localizar los puntos correctos de búsqueda entre imágenes permite un análisis adecuado cuando se presentan efectos de distorsión debidos a movimientos de rotación entre otros. Ellos implementan la técnica de registro en un sistema que funciona bien bajo supervisión humana, es decir, de modo manual. Se captan solamente dos cuadros de imagen y la captura no es simultánea al proceso de cálculo de flujo óptico de tal modo que ellos sugieren como trabajo futuro que el programa funcione de manera automática.

A partir de la publicación del trabajo realizado por Lucas y Kanade en 1981, otros investigadores proponen soluciones para el cálculo de flujo óptico mediante la técnica de registro aportada por dichos autores. Una de estas soluciones es propuesta por Barron, Fleet

y Bauchemin (1994). Ellos implementan un modelo de primer orden para el cálculo de flujo óptico en una secuencia de imágenes donde aplican la técnica de registro de Lucas y Kanade (1981) y la ecuación de establecida por Horn y Schunck (1981). Esta implementación se conoce como Algoritmo de Lucas-Kanade para el cálculo de flujo óptico. En el mismo artículo realizan un reporte de las técnicas más conocidas para el cálculo de flujo óptico. El reporte consiste en comparaciones empíricas de exactitud, fiabilidad y densidad del flujo óptico detectado. Utilizan secuencias de imágenes sintéticas para sus comparaciones.

Los investigadores Simon Baker e Iain Matthews del Instituto de Robótica de la Universidad de Carnegie Mellon (2004), publican una serie de documentos donde comparan las técnicas de Lucas y Kanade contra otros métodos y realizan una implementación utilizando la herramienta de software MATLAB. Esta aplicación no es en tiempo real, el proceso de análisis se efectúa con secuencias almacenadas previamente. Backer y Matthews analizan algunos algoritmos para el procesamiento de la imagen y examinan que extensión de la técnica de Lucas y Kanade es útil sin presentar pérdida considerable de eficiencia. Realizan comparaciones empíricas en cuanto al costo computacional. Al final proponen una clasificación de algoritmos subdividiéndolos en: aditivos, compositivos, directos o inversos. La elección de ellos depende de si hay mayor ruido en la plantilla (directo) o en la imagen (inverso) o si se requiere o no un algoritmo eficiente (inverso compositivo). Los autores no proponen una única solución para la medición de los movimientos.

Existen algunos trabajos doctorales (Barron y Beauchemin, 1995; Zuloaga, 1998), donde la investigación consiste en examinar las diversas aplicaciones de procesamiento para secuencias de imágenes y su relevancia. Estos trabajos analizan las técnicas más comunes para el cálculo de flujo óptico. Mencionan algunos métodos de procesamiento por software para secuencias de imágenes propuestos por diferentes autores y describen las diversas arquitecturas por hardware diseñadas para la detección de movimiento en secuencias de imágenes. Estas investigaciones discuten los métodos y técnicas de extracción de movimiento más conocidas tanto por software como por hardware y no realizan ningún tipo de comprobación o implementación.

Entre los antecedentes más recientes que constatan el interés de los investigadores por determinar el movimiento de escenas dinámicas por medio de sensores visuales, se encuentra otro trabajo que investiga e implementa algoritmos para la detección de movimiento en el campo de visión dinámica por computadora (Kusina, 2000). En esa investigación se describen y analizan algunos métodos de detección de movimiento por medio de gradientes y correspondencias, dos de los cuales se implementan y prueban mediante un software conocido como Halcón 5.0; que es una herramienta útil en el procesamiento de imágenes además de algunas rutinas de proceso en "C". Se emplean imágenes reales (imágenes no sintéticas) de secuencias de sólo dos cuadros, con un análisis fuera de línea, es decir, la secuencia se captura y almacena en archivos, para ser procesada posteriormente. El primer método evaluado por Kusina, está basado en el cálculo de gradientes (Jain, Kasturi y Schunck, 1995). Obtienen los bordes completos de los objetos y después, una imagen sobrepone a la otra para verificar si los objetos se movieron. El segundo método es el algoritmo SUSAN (Smith y Brady, 1997) que detecta las esquinas significativas de los objetos en la escena y al igual que en el primer método, se sobreponen ambas imágenes para ver si la posición de las esquinas cambió. Estos métodos, según los autores del trabajo citado, reducen la cantidad de información a procesar. El primer método genera más información que el segundo pero solo de los bordes. La desventaja del segundo es el problema de escoger un umbral adecuado para cada imagen en particular que depende de la cantidad de esquinas encontradas. En este artículo, proponen como trabajo futuro generar el campo de flujo óptico.

Otro trabajo del área de interés, es la investigación que lleva por título "Arquitectura hardware para la extracción de invariantes ópticos, a partir del flujo óptico, en tiempo real" (Cobos, 2001). En esa tesis, se desarrolla una estructura hardware para calcular el flujo óptico de imágenes reales y generar los invariantes ópticos de un sistema monocular, diseñado para aplicaciones robóticas con visión activa. Esta estructura realiza cálculos de bajo nivel sobre la imagen y aporta información a otros módulos de procesamiento de mayor nivel de complejidad. Este proyecto representa específicamente una etapa de visión para un robot con navegación autónoma. Se utiliza un simulador para comprobar su funcionamiento. Como continuación de este trabajo, se propone la inclusión del módulo de visión en una cabeza robótica con motores para controlar el movimiento de las cámaras con

determinado número grados de libertad. Entre los resultados de la investigación se concreta el correcto funcionamiento modular de los componentes del sistema de visión, y la posibilidad de ser controlados y monitorizados por otro sistema que realice un seguimiento y ajuste del funcionamiento de todos los componentes.

Existen proyectos muy interesantes referentes a vehículos de navegación autónoma, cuyo objetivo es demostrar que sensores de flujo óptico, pueden ser utilizados por un piloto automático para conducir un pequeño vehículo volador conocido como *"aircraft"*, a través de un ambiente real no simulado capaz de evadir obstáculos y controlar altitud mediante el flujo óptico. La revista electrónica *"Wired"* publicó en 2004, un artículo escrito por Lakshmi Sandhana denominado *"Bugs Taking Over Robot Guidance"* (Sandhana, 2004). Este artículo expone sobre los trabajos que se realizan actualmente en varios centros de investigación donde tratan de desarrollar vehículos aéreos no tripulados dotados de un sistema de visión y navegación, inspirados en los insectos. La percepción visual de los insectos está orientada por un fenómeno visual conocido como flujo óptico. Esta investigación se basó en la primera publicación sobre flujo óptico realizada por John Gibson (Gibson, 1979), donde relaciona el concepto de flujo óptico como la forma en que los insectos se guían a través de su medio.

Como se observa, existen muchas investigaciones orientadas a determinar el movimiento en escenas dinámicas, todas con el objetivo de proporcionar nuevas soluciones o mejorar aquellas existentes para el cálculo de movimiento en tiempo real ya que de las soluciones eficientes y efectivas dependen un sin número de aplicaciones útiles para sistemas automatizados de visión. Esta tesis busca contribuir en este sentido.

Para una observación más clara de los antecedentes expuestos, estos se simplifican en la Tabla 1.1

Tabla 1.1. Resumen de antecedentes.

Autor(es)	Año	Nombre de la investigación	Aportación	Resultados
Horn y Schunck	1981	*"Determining Optical Flow"*	Establecen una ecuación para el cálculo de flujo óptico bajo ciertas restricciones. Establecen un método iterativo para la reducción de errores en el cálculo.	El cálculo se basa en dos cuadros de imágenes sintéticas. Obtienen un error máximo de 10% y un mínimo de 1% apliacando 1, 4, 16, 32 y 64 iteraciones a su métdo.

Tabla 1.1. Resumen de antecedentes (continuación).

Autor(es)	Año	Nombre de la investigación	Aportación	Resultados
Lucas y Kanade	1981	*"An Iterative Image Registration Technique with an Application to Stereo Vision"*	Establecen una técnica de registro para el seguimirnto de puntos entre imágenes. Esta técnica es aplicable a la solución de problemas de visión y análisis de movimiento por medio de gradientes.	Realizan pruebas de distancia entre objetos y la cámara en dos cuadros sucesivos de una secuencia obteniendo un error max de .0012% y un mínimo de .0009%. Su implementación es manual. Sugieren como trabajo futuro la funcionalidad automática de su técnica
Barron et al	1994		Aplican la técnica de registro de Lucas y Kanade (1981) a la ecuación establecida por Horn Schunck (1981) obteniendo un modelo de primer orden conocido como Algoritmo de Lucas-Kanade para el cálculo de flujo óptico	Realizan un reporte consistente en comparaciones empíricas de exactitud, fiabilidad y densidad del flujo óptico detectado, utilizando secuencias de imágenes sintéticas para sus comparaciones y calculan los errores de cada técnica
Barron y Bauchemin Zuloaga	1995 1998	*"The Computation of Optical Flow"* *"Visión Artificial Dinámica"*	Ambos trabajos examinan diversos métodos y técnicas para la detección de movimiento en una secuencia de imágenes. Describen tanto los métodos de procesamiento por software como por hardware y su relevancia en algunas aplicaciones	Estos trabajos discuten los métodos y técnicas de extracción de movimiento, tanto por software como por hardware y no realizan ningún tipo de comprobación o implementación.
Kusina	2000	*"Visión dinámica por computadora"*	En este trabajo se describen e implementan 2 algoritmos de detección de movimiento: Algoritmo de Jain y el algoritmo de Susan	Se realizan pruebas con secuencias de 2 cuadros de imagen previamente almacenadas, se comparan ambos métodos y se describen sus diferencias, se propone como trabajo futuro generar el campo de flujo óptico
Cobos	2001	*"Arquitectura Hardware para la extracción de invariantes ópticos a partir del flujo óptico en tiempo real"*	Desarrolla una estructura hardware para calcular el flujo óptico en imágenes reales para aplicaciones robóticas de visión activa	Se obtiene una etapa de visión para un robot con navegación autónoma. Este proyecto no es implementado. Se utiliza un simulador para su comprobación. Se propone como trabajo futuro la inclusión del módulo en una cabeza robótica con motores para controlar el movimiento de las cámaras.

Tabla 1.1. Resumen de antecedentes (continuación).

Autor(es)	Año	Nombre de la investigación	Aportación	Resultados
Revista electrónica "Wired" Shandhana	2004	*"Bugstaking Over Robot Guidance"*	Investigación sobre vehículos aéreos no tripulados controlados por flujo óptico. Proyecto inspirado en la visión de algunos insectos que utilizan el flujo óptico para guiarse en su medio.	Se realizan pruebas con pequeños aviones conocidos como "aircraft" capaces de evadir obstáculos y controlar altitud por medio del flujo óptico. La implementación del sistema de control es por hardware.

1.4 Objetivo general

El objetivo de esta tesis es detectar el movimiento a partir de la información obtenida de una secuencia de imágenes, mediante la implementación de algoritmos para el cálculo de flujo óptico, ejecutados en tiempo real, utilizando una computadora personal y sensor de video de propósito general.

1.5 Objetivos específicos

- Implementar los algoritmos para cálculo de flujo óptico por medio de software.
- Desarrollar un sistema que procese una secuencia de imágenes en tiempo real y que proporcione como resultado la detección de movimiento de manera simultánea al procesamiento.
- Conseguir la funcionalidad del sistema como etapa previa de procesamiento para aplicaciones de visión por computadora que requieren la utilización de parámetros de movimiento.
- Obtener un sistema transportable y aplicable a diversas plataformas.
- Establecer un procedimiento general para la implementación por software de algoritmos para el cálculo de flujo óptico.

1.6 Resumen de la metodología aplicada

Para la consecución de los objetivos trazados se planteó la siguiente metodología. Como primer paso se realizó una revisión bibliográfica sobre el tema de visión dinámica. La visión dinámica es un tema extenso de estudio. Dentro de esta área se encontró un tema conocido como flujo óptico: fenómeno mediante el cual el movimiento es detectado visualmente por los animales. Este tema resulta ser muy interesante en cuanto al potencial de aplicaciones donde la detección del movimiento es el principal objetivo, como las observaciones meteorológicas, sistemas de vigilancia por cámara y sistemas de navegación controlados por visión. Este potencial de aplicaciones, motivaron esta investigación y el planteamiento del problema.

Cómo paso subsecuente, se analizan antecedentes de las técnicas y métodos para el cálculo de flujo óptico en escenas dinámicas con la finalidad de seleccionar aquellos que presentan mejores resultados en sus aplicaciones. Se obtienen finalmente dos algoritmos básicos conocidos como Algoritmo de Horn-Schunck (Horn y Schunck, 1981) y Algoritmo de Lucas-Kanade (Lucas y Kanade, 1981). A partir de este análisis se establece el objetivo de este trabajo, cuyo desarrollo se dividió en tres etapas.

La primera etapa es de análisis y prueba; consiste en realizar programas para ambos algoritmos utilizando la herramienta MATLAB versión 6.0 con la finalidad de establecer el costo computacional de cada uno para cálculo de flujo óptico.

La segunda etapa consiste en modelar un sistema de cálculo de flujo óptico en diagramas de bloque con la herramienta *"Video and Image Processing Blockset"* de *Simulink*, con el fin de verificar su funcionalidad en tiempo real y analizar algunos parámetros en cada algoritmo como la sensibilidad al ruido, la detección de áreas falsas de cálculo, el tiempo de ejecución en cuadros por segundo y la posibilidad de adaptación a diversos escenarios (Diferente tipo de iluminación, objetos que oculten o descubran a otros objetos). El modelo se basa en las fases para el cálculo de flujo óptico por gradientes en una secuencia de imágenes, establecidas por Horn y Schunck (1981).

En la tercera y última etapa de la implementación se selecciona uno de estos dos algoritmos en base a los datos obtenidos en las etapas anteriores. Se modifican las fases de cálculo de flujo óptico por gradientes para una secuencia de imágenes, reduciendo el costo computacional. Se añade como parte de las fases, un algoritmo para la localización de puntos característicos desarrollada por Shi y Tomasi, (1994) que aumenta la funcionalidad del algoritmo para cálculo de flujo óptico debido a que se efectúa solo en áreas adecuadas o seleccionadas. Se desarrolla también, una función propia para generar y desplegar gráficamente, el campo de flujo óptico. El programa final se desarrolla en Lenguaje "C'', por ser funcional en diversas plataformas logrando con ello un software transportable.

Por último, se realizan algunas aplicaciones utilizando las funciones desarrolladas para él cálculo de flujo óptico. Una de las aplicaciones consiste en emplear el campo de flujo óptico generado, para la localización y seguimiento de rostros, logrando con ello un seguimiento continuo sin perder el área localizada. La segunda aplicación es el uso en un instrumento para análisis cuantitativo del movimiento de microestructuras, para la determinación de planos focales en un microscopio donde la estabilización de los vectores de flujo óptico, coincide con el enfoque del lente, logrando con ello un enfoque automático.

1.7 Importancia del estudio

El lograr implementar métodos y técnicas de análisis de movimiento para una secuencia de imágenes en tiempo real por medio de software utilizable en cualquier plataforma, sin la necesidad de hardware específico o externo, es la pauta para el diseño de sistemas versátiles controlados por visión. Este trabajo aporta un sistema por software para el cálculo de flujo óptico donde: los parámetros resultantes del análisis (magnitud y dirección de movimiento) en una secuencia de video, son aplicables al desarrollo de proyectos controlados por visión. Otra de las contribuciones es un cambio propuesto en las fases para la implementación de algoritmos por software para el cálculo de flujo óptico que reduce el costo computacional**.**

1.8 Limitaciones del estudio

- No se calcula el flujo óptico si la cámara esta frente a una superficie completamente lisa, ya que ésta es una limitación propia del fenómeno de flujo óptico.

- No se realiza el almacenamiento físico, simultáneo al proceso de cálculo de flujo óptico puesto que retarda el tiempo de ejecución del programa. Se almacenan resultados en un arreglo temporal que al final puede enviarse a un archivo físico.

1.9 Estructura del documento

El documento que integra la presente investigación está constituido por cuatro capítulos organizados de la siguiente manera:

En el capítulo II se presenta el marco teórico donde se definen algunos conceptos de importancia relacionados con el tema de la investigación así como los métodos para la detección de movimiento, subrayando las ventajas y desventajas implícitas en la aplicación de cada uno de ellos, mencionando a su vez los algoritmos que utilizan estos métodos en el análisis de movimiento en una secuencia de imágenes. Se describen las fases para el cálculo de flujo óptico y los algoritmos de análisis de movimiento implementados en este trabajo. También se hace una descripción de las plataformas de desarrollo utilizadas en la implementación de los algoritmos para el cálculo de flujo óptico.

En el capítulo III se describe el desarrollo del proyecto en tres etapas donde la primera etapa consiste en el desarrollo de programas para la comprobación de las diferentes técnicas y métodos basados en gradientes, utilizados en la detección de movimiento en escenas dinámicas. En esta etapa se utiliza MATLAB 6.1 como herramienta de desarrollo. La segunda etapa es el diseño de un modelo aproximado para detección de movimiento en tiempo real, utilizando módulos de procesamiento de imágenes y captura de video, mediante la herramienta *"Video and Image Processing Blockset"* de Simulink, que es parte del software MATLAB versión 2006a. Se establece también un procedimiento de implementación, en base a resultados heurísticos obtenidos. En la última etapa se desarrolla

el sistema utilizando la biblioteca de funciones conocida como OpenCV de la compañía Intel, el lenguaje de programación “C” y el compilador de Visual C++.

El último capítulo presenta los resultados y conclusiones obtenidos en cada etapa de implementación y desarrollo del proyecto. En este capítulo se exponen también los resultados obtenidos a través de la utilización del sistema desarrollado, como etapa previa para detección de movimiento, en dos aplicaciones específicas creadas con la finalidad de comprobar su funcionalidad y portabilidad como etapa de proceso en aplicaciones que requieren de la detección de movimiento. Para terminar el capítulo se discuten los resultados y se enfatizan las aportaciones del trabajo realizado.

Al final del documento se incluyen en apéndices los códigos de los programas que se desarrollan en este trabajo, para consulta del lector. También se anexa un glosario con la definición de conceptos importantes en esta tesis y la lista de los productos obtenidos durante la investigación.

CAPÍTULO II

TÉCNICAS PARA LA DETECCIÓN DE MOVIMIENTO Y CÁLCULO DE FLUJO ÓPTICO

En este capítulo se presentan las bases para el cálculo de flujo óptico en una secuencia de imágenes, y se definen algunos conceptos de importancia relacionados con el tema. Una lista más completa de definiciones se presenta en el glosario incluido al final del documento. Se describen los métodos para la detección de movimiento, subrayando las ventajas y desventajas implícitas en la aplicación de cada uno de ellos, mencionando a su vez los algoritmos que utilizan estos métodos en el análisis de movimiento para una secuencia de imágenes. Se hace también una descripción de las fases para el cálculo de flujo óptico y los algoritmos de análisis de movimiento implementados en este trabajo. Por último se describen brevemente las plataformas de desarrollo utilizadas para la implementación de los algoritmos que determinan el flujo óptico.

2.1 Imagen digital

Las imágenes digitales son el principal elemento de lo que se conoce como visión por computadora y representan una escena del entorno mediante algún tipo de codificación, normalmente como una matriz de números de dos dimensiones a la que se llama, arreglo bidimensional de píxeles con diferente intensidad luminosa. Diferentes tipos de imágenes digitales se definen dependiendo de las características de ésta; se verifica el glosario para los conceptos utilizados en este documento.

2.2 Flujo óptico

El término de flujo óptico se refiere a un fenómeno visual que un observador que se desplaza, experimenta al percibir el movimiento aparente de los objetos frente a su campo visual, tal como se muestra en la Figura 2.1. Por ejemplo, el movimiento percibido por el observador en dirección al suelo es en sentido contrario a su trayectoria y el flujo óptico percibido al acercarse a un objeto estático se dispersa alrededor de un punto central referido por el objeto móvil. Los objetos más próximos desaparecen de su

campo visual a una velocidad mayor que los más lejanos. Por medio de este movimiento aparente, el observador determina que tan cerca o que tan lejos se encuentra de los objetos que observa. Este concepto fue estudiado por Euclides en su tratado de Óptica allá hacia el año 300 a. de C. (Martinez, 2002) como rayos visuales con los cuales obtiene características espaciales de los objetos a distancia y considerado como un fenómeno de percepción visual por Gibson (1979).

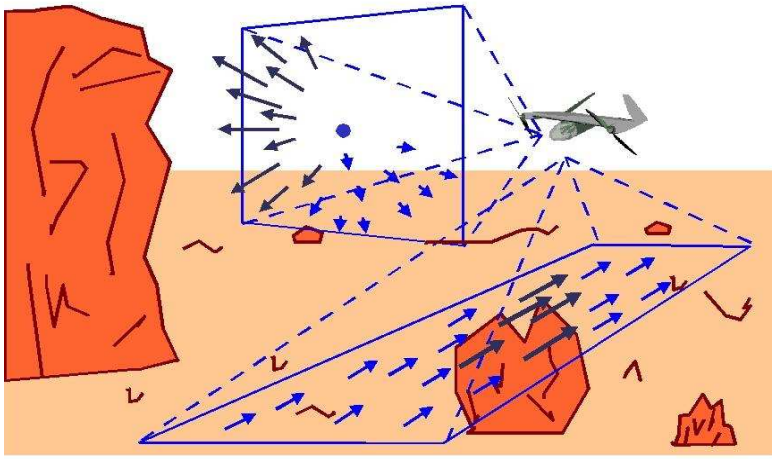


Figura 2.1. Campo de flujo óptico percibido por un objeto móvil.

La noción de movimiento relativo también se determina en una secuencia de imágenes digitales captada por una cámara en movimiento o por un sensor estático donde los objetos son los que se mueven. La definición de flujo óptico para una secuencia de imágenes digitales varía un poco con respecto al fenómeno visual descrito, pero en esencia ambos permiten determinar el movimiento aparente de los objetos.

2.3 Flujo óptico en imágenes digitales

El flujo óptico se define como el movimiento aparente de objetos en una secuencia de imágenes digitales, debido a cambios en la intensidad de los píxeles. La intensidad de estos varía dependiendo de otros factores que como el cambio de la iluminación, no son el movimiento de objetos. En el caso ideal en el que estas variables sean constantes, un punto de la imagen conserva su intensidad al moverse, así que se dice que los cambios de intensidad en la imagen son debidos al movimiento de un punto, y entonces considerar el flujo óptico como el valor de la velocidad de ese punto. Según diversos autores, el flujo óptico es la velocidad aparente de las estructuras de niveles de gris (Otte y Nagel, 1995), resultando un arreglo bidimensional de vectores. El flujo óptico se

considera como una aproximación del campo de movimiento (cambio de posición de las partículas que conforman un objeto, con respecto al tiempo); en una secuencia de imágenes no existe el cambio de partículas pero si el cambio de intensidad con respecto al tiempo (Gupta y Kanal, 1995). Por consiguiente, las variaciones en los niveles de gris son debidas al movimiento de los objetos, por ello a partir del flujo óptico es posible determinar el movimiento de los objetos en secuencias de imágenes (Horn y Schunck, 1981).

2.4 Métodos de detección de movimiento en imágenes digitales

La formulación de algoritmos eficientes para calcular el movimiento y la estructura de los objetos a partir de secuencias de imágenes, ha sido una de las áreas de mayor interés en el campo de la visión artificial. Se han propuesto una gran cantidad de algoritmos para la detección del movimiento en imágenes de video. Cada autor establece algunas mejoras o nuevas formas de resolver el problema del cálculo del flujo óptico en una serie de imágenes. Básicamente se utilizan dos formas para su solución, con diversas variaciones sobre ellos. Estas formas son la segmentación de imágenes y el método de gradientes espacio-temporales.

2.4.1 Segmentación de las imágenes

El método de segmentación consiste en identificar una serie de características locales (objetos, bloques, segmentos, etc.) en una imagen, tratar de obtener las correspondencias con otra imagen, y por último, determinar los movimientos. Estas técnicas no generan un campo vectorial muy denso, puesto que los puntos que representan los vectores corresponden solamente a ciertas áreas seleccionadas de la imagen. Cuando una imagen está compuesta por múltiples objetos, las segmentaciones suelen ser numerosas; por tanto, el cálculo es extremadamente laborioso por lo que las técnicas y métodos de segmentación son computacionalmente muy exigentes, pero no están limitadas a ciertas restricciones que ocurren con el segundo método basado en el cálculo de gradientes (Schweitzer, 1995). Muchos autores resuelven el problema de determinación del movimiento separando la imagen en partes y luego determinando el movimiento de cada una de estas partes, aunque esto implica un procesamiento lento y difícil de implementar por software en tiempo real. Además, exige el tener

conocimiento sobre los objetos en la escena ya que regularmente la segmentación es utilizada para la determinación de las características del movimiento. En este tipo de técnicas existen tres algoritmos básicos para segmentación de imágenes:

- **Algoritmos de segmentación en líneas.** Estos algoritmos extraen el contorno de los objetos, los modelan como segmentos de líneas, y determinan los parámetros de movimiento en base al desplazamiento de los segmentos entre las imágenes sucesivas. Ejemplos de éstos algoritmos se pueden encontrar en (Gupta, Kanal, y Zhan, 1995).

- **Algoritmos de segmentación en bloques.** Estos modelan las imágenes dividiéndolas en rectángulos (u otros polígonos) de tamaños variables que se adaptan a áreas de nivel de gris (o color) más o menos homogéneos. El movimiento se determina en base al desplazamiento de los centros de los rectángulos entre las sucesivas imágenes. Ejemplos de estos algoritmos se encuentran en (Panjwani, Healey y Schweitzer, 1995).

- **Algoritmos de segmentación de grupos (*clusters*).** Modelan las imágenes como áreas amorfas adaptadas a áreas de la imagen de características más o menos homogéneas en cuanto a color, nivel de gris o textura. El movimiento se determina en base al desplazamiento de los centros de masa de cada una de las áreas correspondientes entre dos imágenes consecutivas. Ejemplos de estos algoritmos se encuentran en (Kottke, Sun, Pardàs, Salembier, Uchiyama y Arbib, 1994).

2.4.2 Método de gradientes espacio-temporales

El método de gradientes espacio-temporales consiste en determinar los cambios (gradientes) espaciales y temporales de los niveles de gris de la imagen y a partir de ellos, obtener el flujo óptico. Este método tiene como base el hecho de que en los cambios de intensidad de la imagen está contenida la información del movimiento de los objetos. Tiene la ventaja de que genera un campo vectorial muy denso (un vector por píxel) que resulta muy útil en aplicaciones donde el interés es la detección de movimiento y no tanto la detección de objetos y sus características. Ya que este método se basa en determinar cambios de intensidad entre las imágenes, su costo computacional

resulta mínimo para aplicaciones de cálculo de flujo óptico en tiempo real. Sin embargo, presenta el problema de ser muy sensible al ruido en las imágenes y no determina el flujo óptico bajo ciertas circunstancias como la imposibilidad de encontrar el movimiento de un punto correspondiente a un borde rectilíneo o a un área uniforme de un cuadro a otro en una secuencia de imágenes, cuando las dimensiones de dicho borde o área abarcan el espacio completo de la ventana que se observa. Lo anterior se conoce como "problema de la apertura". Para que este método encuentre la correspondencia de un punto entre dos imágenes consecutivas, el punto debe ser diferente de sus vecinos inmediatos. Algunos algoritmos que emplean este método utilizan umbrales como solución al problema. Por otro lado, los métodos basados en cálculo de gradientes están limitados por ciertas restricciones. Así, el objeto que se mueve no cambia de forma y no detecta el movimiento de objetos que se mueven detrás de otros, además del requerimiento de una iluminación constante (Laplante y Stoyenko, 1996). Para cumplir las restricciones de este método es necesario llevar a cabo una serie de fases que permiten la extracción adecuada del flujo óptico por gradientes.

2.5 Fases para el cálculo de flujo óptico por gradientes

Las fases para el cálculo de flujo óptico por gradientes consisten en una serie de pasos básicos, aplicados a una secuencia de imágenes. Horn y Schunck (1981) establecen cuatro fases para la estimación del flujo óptico, como se observa en la Figura 2.2:

- Suavizado de la secuencia.
- Cálculo de gradientes.
- Determinación del flujo óptico básico.
- Aplicación de restricciones de suavidad en cambios en el flujo óptico.

Yamamoto (1989) establece un procedimiento de tres fases aplicadas a la secuencia de imágenes para el cálculo de flujo óptico en parámetros de movimiento 3-D, tal como lo ilustra la Figura 2.3, similar al procedimiento anterior pero sin incluir la aplicación de restricciones de suavidad.

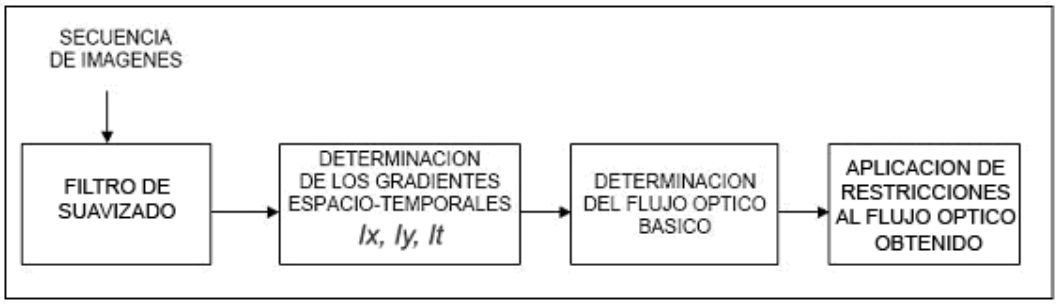


Figura 2.2. Fases para la determinación del flujo óptico en una secuencia de imágenes establecidas por Horn-Schunck.

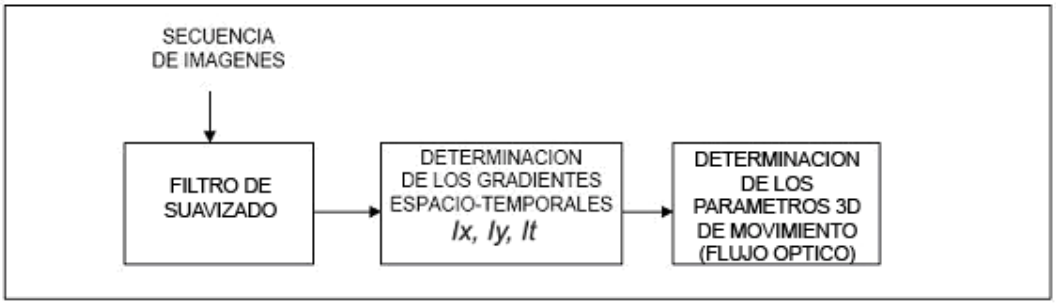


Figura 2.3. Fases para la determinación de los parámetros de movimiento 3-D del flujo óptico en una secuencia de imágenes establecidas por Yamamoto.

2.5.1 Filtro de suavizado

La primera fase, establecida por ambos autores, consiste en aplicar filtros de suavizado sobre la secuencia de imágenes para disminuir el ruido y evitar cambios bruscos en el movimiento entre puntos cercanos en la imagen. El tipo de filtros a utilizar en esta fase depende del autor del algoritmo implementado o del entorno de la escena. Para comprender en qué consiste el filtrado de una imagen, se definen algunos conceptos relacionados y se describen los filtros (operadores) más utilizados en los algoritmos de cálculo de flujo óptico por gradientes.

La convolución es la base de algunos procesamientos comunes, como el filtrado de imágenes y se utiliza para la evaluación de gradiente, el promediado de nivel de gris y la búsqueda de contornos entre otras. En la convolución, el valor de un píxel de salida $G[x,y]$ se calcula mediante la suma ponderada los píxeles vecinos (ver Figura 2.4). Dentro del campo de procesamiento de imágenes, la convolución se realiza entre la imagen original $F[x,y]$ y una matriz $H[x,y]$ (los coeficientes del filtro) llamada máscara para filtrar una imagen (Jain *et al.*,1995; Pajares y de la Cruz, 2002). Matemáticamente la convolución es representada como:

$$G[x,y] = F[x,y] * H[x,y] = \sum_j \sum_k F[x-j, y-k] * H[j,k] \qquad (2.1)$$

La matriz operador $H[x,y]$ por lo general tiene dimensiones de 3x3 a 5x5 elementos y su tamaño repercute en la calidad del filtro. Estas matrices operadores se observan en la Figura 2.5.

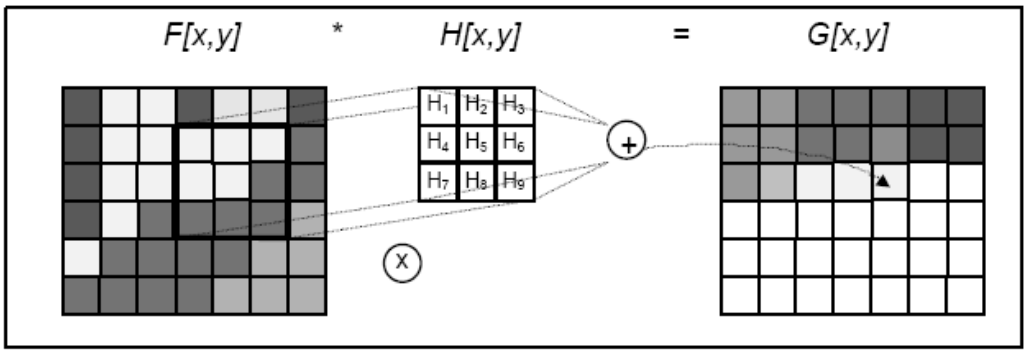


Figura 2.4. Representación gráfica de la convolución.

Para aplicaciones, en las cuales intervienen secuencias de imágenes, es posible definir una convolución tridimensional, con dos dimensiones espaciales y una temporal:

$$G[x,y,t] = F[x,y,t] * H[x,y,t] = \sum_i \sum_j \sum_k F[x-i, y-j, t-k] * H[i,j,k] \qquad (2.2)$$

Filtrado de una imagen. El filtrado es una técnica para modificar o mejorar a una imagen. Por ejemplo, un filtro resalta o atenúa algunas características. El filtrado es una operación de vecindad, en la cual el valor de un píxel dado en la imagen procesada, se calcula mediante un algoritmo que toma en cuenta los valores de los píxeles en la vecindad de la imagen original. Los filtros son utilizados para eliminar el ruido de una imagen, para destacar los bordes de un objeto y para detección de contornos. Entre los filtros (operadores) más usados, que además permiten un suavizado del gradiente y detectan contornos se encuentran: Roberts, Prewitt, Sobel, isotrópico y Laplaciano (Jain *et al.*,1995; Pajares y de la Cruz, 2002) (ver Figura 2.5).

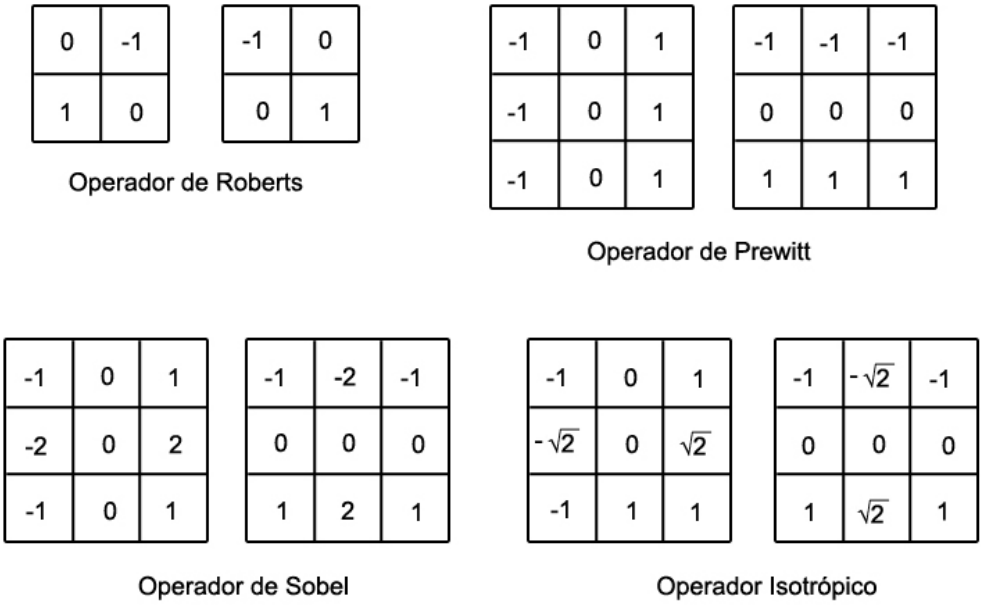


Figura 2.5. Operadores más utilizados para suavizado de gradientes y detección de contornos.

En una imagen los contornos corresponden a los límites de los objetos presentes en la misma. Para encontrar los contornos se buscan los lugares en la imagen, en los cuales la intensidad del píxel cambia rápidamente, por lo general usando alguno de los siguientes criterios (Jain *et al.*,1995; Pajares y de la Cruz, 2002):

- Lugares donde la primera derivada (gradiente) de la intensidad es mayor que el umbral predefinido (Roberts, Prewitt, Sobel e isotrópico).
- Lugares donde la segunda derivada (Laplaciano) tiene un cruce por cero.

En el primer caso se buscan grandes picos, y en el segundo, cambios de signo, como se muestra en la Figura 2.6.

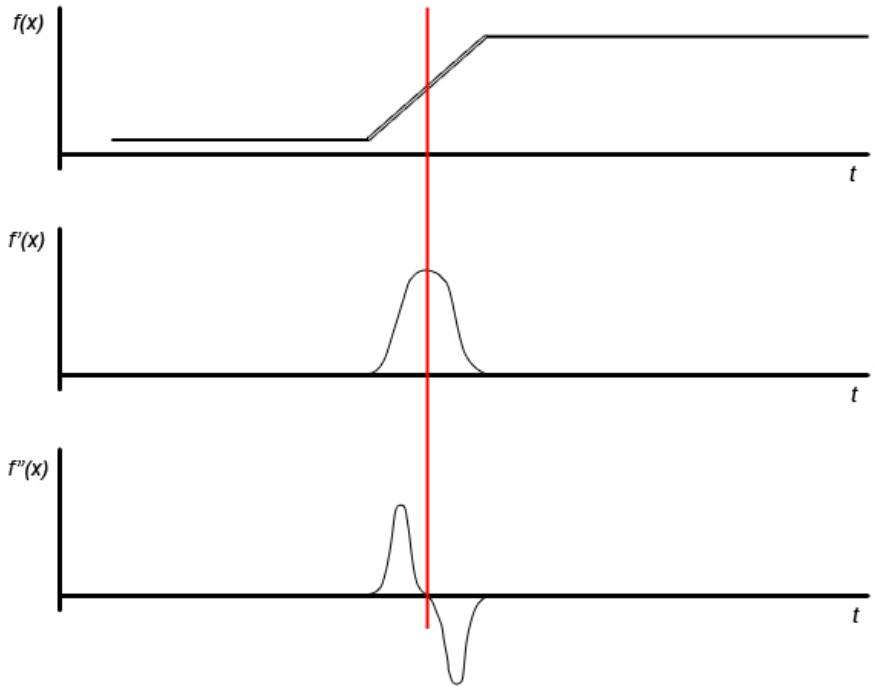


Figura 2.6. Criterios de la primera y segunda derivada de una función.

Estos criterios consisten en destacar los bordes (cambios de iluminación) de la imagen usando un filtro basado en gradientes. La detección de bordes es esencialmente la operación de detectar cambios locales significativos en una imagen. El gradiente es una medida de cambio en una función de intensidad. Los cambios significativos en valores de gris en una imagen, se destacan utilizando aproximaciones discretas del gradiente (Jain *et al.*,1995; Pajares y De la Cruz, 2002).

El gradiente de una imagen $f(x,y)$ es el valor

$$G[f(x,y)] = \begin{bmatrix} G_x \\ \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{2.3}$$

que indica la dirección de la máxima variación de f en el punto (x,y). El valor del vector es su módulo que es igual a la máxima variación de $f(x,y)$ por unidad de

distancia en la dirección del gradiente. Entonces el valor absoluto del gradiente se calcula de acuerdo a las fórmulas.

$$G[f(x,y)] = \sqrt{G_x^2 + G_y^2} \tag{2.4}$$

$$G[f(x,y)] \approx |G_x| + |G_y| \tag{2.5}$$

$$G[f(x,y)] \approx \max(|G_x|, |G_y|) \tag{2.6}$$

Es habitual aproximar la magnitud del gradiente con valores absolutos (2.5) y (2.6), porque reduce el costo computacional y el valor de la magnitud del gradiente no es tan importante cuando se decide si un punto pertenece o no a un borde mediante la comparación de un umbral.

La dirección del vector del gradiente se identifica por el ángulo

$$\alpha(x,y) = \tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{2.7}$$

Entonces el cálculo de gradiente de una imagen se basa en la obtención de las derivadas parciales de $\partial f / \partial x$ y $\partial f / \partial y$ en cada posición del píxel. Las derivadas se implementan en forma digital usando máscaras de convolución (Figura 2.5).

2.5.2 Determinación de los gradientes espacio-temporales

En la segunda fase del cálculo de flujo óptico por gradientes, tanto Horn y Schunck (1981) como Yamamoto (1989), determinan los gradientes espacio-temporales utilizando una ecuación que relaciona las velocidades de los puntos en la escena con la variación del nivel de gris en la imagen. Esta ecuación es conocida como la ecuación restringida de flujo óptico *(Optical Flow Constraint Equation o Spatio-Temporal Constraint Equation)* y es formulada por Berthold K.P. Horn y Brian G. Schunck (1981):

$$I_x V_x + I_y V_y + I_t = 0 \tag{2.8}$$

donde la intensidad en un punto de la imagen es $I(x,y,t)$; I_x e I_y representan los gradientes espaciales, I_t el gradiente temporal; V_x y V_y son los vectores de velocidad horizontal y vertical respectivamente del flujo óptico. Con esta ecuación, se conoce solamente una componente de la velocidad. Para hallar la segunda componente de la velocidad deben hacerse suposiciones adicionales las cuales varían dependiendo del autor del algoritmo.

Yamamoto (1989) aplica esta ecuación restringida de flujo óptico, para calcular parámetros 3-D del movimiento de un objeto rígido, agregando la información de profundidad del objeto y la ecuación vectorial de velocidad (2.9) en la ecuación restringida para el cálculo de flujo óptico (2.8).

$$V = \Omega \times r + V_L, \tag{2.9}$$

donde Ω es el vector velocidad angular $(\omega_x \omega_y \omega_z)$, r es la posición del punto (x,y,z) en la imagen y V_L es el vector velocidad en el punto (v_x, v_y, v_z).

A partir de las ecuaciones (2.8) y (2.9) se obtiene la ecuación siguiente para el caso tridimensional,

$$I_x V_x + I_y V_y - I_y \omega_x z + I_x \omega_y z + (I_y x - I_x y)\omega_z + I_t = 0 \tag{2.10}$$

donde z es conocida.

El interés de esta tesis no es el cálculo de los parámetros 3-D, puesto que no se calcula ni se conoce al objeto, solo se requiere el cálculo del flujo óptico simple (2-D), el cual se obtiene de la ecuación restringida del flujo óptico (2.8). Para la aplicación de esta ecuación se debe considerar una secuencia de imágenes, donde se cumplen las siguientes condiciones:

- La iluminación de los objetos presentes en las imágenes es uniforme, la reflexión de los objetos varía con suavidad y sin discontinuidades espaciales.
- No existen objetos que se oculten unos detrás de otros.
- No aparecen nuevos objetos en la secuencia de imágenes.

Cada punto en la imagen $I(x,y,t)$ es producido por la proyección de un punto-objeto del espacio en el plano de la imagen y, según las condiciones anteriores, se afirma que la intensidad de cada punto-objeto en particular no cambia, lo que cambia es su posición:

$$I(x,y,t) = I(x+\delta_x, y+\delta_y, t+\delta_t). \tag{2.11}$$

Aquí δ_x y δ_y son los desplazamientos en las direcciones x e y y δ_t es el cambio en el tiempo. Se aplican en la ecuación (2.11) las series de Taylor solo en su primera derivada ($f(a+h) \approx f(a) + f'(a)h$); como se busca una aproximación lineal, las derivadas de segundo orden y superiores se consideran despreciables y se obtiene:

$$I(x,y,t) \approx I(x,y,t) + \delta_x \frac{\partial I}{\partial x} + \delta_y \frac{\partial I}{\partial y} + \delta_t \frac{\partial I}{\partial t} + \varepsilon\,. \tag{2.12}$$

Si se substrae $I(x,y,t)$ en ambos lados de la ecuación y se divide por δ_t, se tiene:

$$\frac{\delta_x}{\delta_t}\frac{\partial I}{\partial x} + \frac{\delta_y}{\delta_t}\frac{\partial I}{\partial y} + \frac{\partial I}{\partial t} + 0(\partial t) \approx 0\,. \tag{2.13}$$

Si el límite δ_t tiende a $0(\delta_t \to 0)$, se obtiene la ecuación restringida del flujo óptico:

$$\frac{dx}{dt}\frac{\partial I}{\partial x} + \frac{dy}{dt}\frac{\partial I}{\partial y} + \frac{\partial I}{\partial t} = 0 \tag{2.14}$$

Esta ecuación se obtiene también de acuerdo a la regla de la cadena y se expresa en múltiples formas como se muestra a continuación.

Se asignan:

$$I_x = \frac{\partial I}{\partial x}, \quad I_y = \frac{\partial I}{\partial y}, \quad I_t = \frac{\partial I}{\partial t}. \tag{2.15}$$

y

$$u = \frac{dx}{dt}, \qquad v = \frac{dy}{dt}. \tag{2.16}$$

Se tiene entonces una ecuación lineal con dos incognitas:

$$I_x u + I_y v + I_t = 0, \tag{2.17}$$

donde u y v son las velocidades en cada dirección.

Otra expresión es

$$(I_x, I_y).(u,v) + I_t = \nabla I \cdot V + \frac{\partial I}{\partial t} = 0, \tag{2.18}$$

donde:

V es el vector de velocidad (u,v) y

∇I es el vector gradiente espacial del nivel de gris $\left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right)$ (2.19)

Por lo tanto, la componente de la velocidad en un punto de la escena, en la dirección del gradiente de intensidad es:

$$(u,v) = -\frac{I_t}{\sqrt{I_x^2 + I_y^2}} \tag{2.20}$$

Sin embargo, no es posible determinar en ciertas zonas la componente del flujo óptico perpendicular al gradiente de intensidad, y tampoco es posible determinarse en las zonas de igual intensidad. **En la Figura 2.7, l**a gráfica indica que la velocidad puede

estar en cualquier punto de la recta, por lo tanto la componente del flujo óptico perpendicular al gradiente de intensidad no puede ser determinada. Esto se conoce con el nombre de problema de apertura.

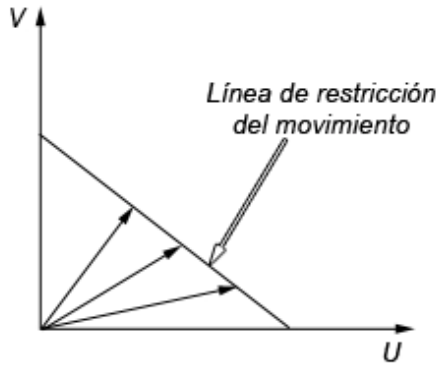


Figura 2.7. Problema de la apertura.

Por ejemplo, no es posible encontrar un punto perteneciente al contorno en la siguiente imagen de una secuencia, a menos que en la zona o apertura se encuentre una esquina u otra componente con suficiente estructura en la intensidad que permita el cálculo de las dos componentes del flujo óptico. En la figura 2.8 se observan los casos del problema de la apertura. En los casos a y b es imposible distinguir el movimiento de un punto observado a través de una pequeña ventana. En el caso c se puede calcular el vector de desplazamiento sin ambigüedad en una esquina.

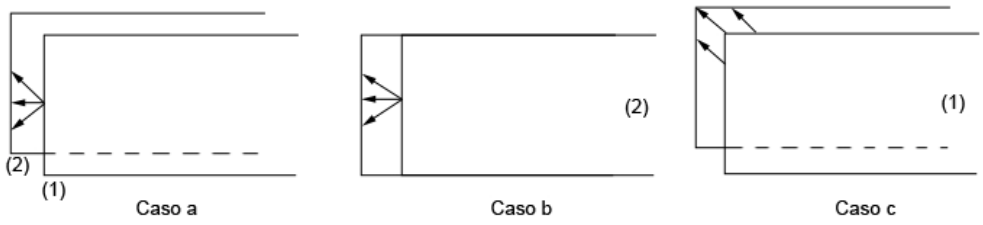


Figura 2.8. Casos del problema de la apertura.

Este tipo de restricciones o errores son resueltos de distintas formas dependiendo del algoritmo seleccionado para la determinación del flujo óptico. Estos algoritmos se describen en la siguiente sección.

2.5.3 Determinación del flujo óptico básico

Como tercera fase en los procedimientos de Horn-Schunck y Yamamoto, se determinan las componentes de velocidad del flujo óptico. El primer autor determina el flujo óptico básico y el segundo calcula los parámetros de movimiento 3-D. Como se menciona en el objetivo general de esta tesis, este trabajo está enfocado a la detección de los parámetros del flujo óptico básico. Para la determinación de estas componentes se aplican algoritmos basados en gradientes. Existen algunos algoritmos para la detección de movimiento en secuencias de imágenes que requieren obtener el flujo óptico, y gran parte de ellos utiliza la ecuación restringida del flujo óptico establecida por Horn y Schunck (1981). Cada algoritmo propone una forma distinta para determinar el valor del flujo óptico donde la ecuación falla. Por consiguiente se describen las técnicas para cálculo del flujo óptico mediante los dos algoritmos más representativos del método de gradientes y una variante del segundo algoritmo. Estos algoritmos se implementan en este trabajo de tesis.

2.5.3.1 Algoritmo de Horn-Schunck

El primero de los algoritmos a describir es el propuesto por Horn y Schunck (1981). Es un método que utiliza las ecuaciones diferenciales de primer orden. Incluye una restricción adicional conocida como continuidad de la velocidad. Realiza una minimización de errores de conservación (intensidad no constante) y de propagación (suavidad global). Los autores asumen que los cambios de intensidad en toda la imagen son suaves y calculan una estimación del campo de velocidad $[u \quad v]^T$, la cual reduce la contribución de errores (2.21).

$$\varepsilon = \iint (I_x u + I_y v + I_t)^2 \, dxdy + \alpha \iint \left\{ \left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial u}{\partial y} \right)^2 + \left(\frac{\partial v}{\partial x} \right)^2 + \left(\frac{\partial v}{\partial y} \right)^2 \right\} dxdy \,. \quad (2.21)$$

Aquí $\frac{\partial u}{\partial x}$ y $\frac{\partial u}{\partial y}$ son las derivadas espaciales de la componente de velocidad horizontal u del flujo óptico, $\frac{\partial v}{\partial x}$ y $\frac{\partial v}{\partial y}$ son las derivadas espaciales de la componente de velocidad vertical v y α es el termino de suavidad global que pondera el error de propagación. La

ecuación (2.21) se compone de la minimización de errores de conservación de la información y de propagación.

Conservación de la información. La conservación de la información se refiere a obtener toda la información disponible localmente de alguna propiedad invariante en la imagen (nivel de gris, derivada del nivel de gris). La conservación de la información se define con la siguiente ecuación, la cual obtiene únicamente la velocidad normal.

$$\varepsilon_b^2 = (I_x u + I_y v + I_t)^2 \tag{2.22}$$

Propagación de la información. En algunas zonas solo se obtiene información parcial sobre la velocidad o simplemente no se obtiene información (problema de la apertura). La información se debe propagar de zonas de gran confianza (las características de la zona analizada permiten obtener vectores de velocidad) a zonas de baja confianza (imposibilidad de obtener parámetros de velocidad por las características de la zona analizada). La información se propaga con técnicas de suavizado. En este algoritmo se utiliza el suavizado global que consiste en la adaptación de la velocidad minimizando una función definida para toda la imagen. Esto significa que el flujo óptico varía lentamente.

Propagación de la información,

$$\varepsilon_c^2 = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \tag{2.23}$$

Simplificando la ecuación (2.21) con las ecuaciones (2.22) y (2.23) se obtiene la siguiente función de error:

$$\varepsilon^2 = \iint (\alpha^2 \varepsilon_c^2 + \varepsilon_b^2) dx dy \,. \tag{2.24}$$

Para obtener el campo de velocidad $\begin{bmatrix} u & v \end{bmatrix}$ para cada píxel en la imagen, se aplica cálculo de variaciones para resolver la minimización de errores obteniendo:

$$(\alpha^2 + I_x^2 + I_y^2)(u - \bar{u}) = -I_x[I_x \bar{u} + I_y \bar{v} + I_t] \tag{2.25}$$

$$(\alpha^2 + I_x^2 + I_y^2)(v - \bar{v}) = -I_y[I_x \bar{u} + I_y \bar{v} + I_t] \tag{2.26}$$

Aplicando él método de minimización de errores de Horn y Schunck , se obtiene un par de ecuaciones para cada punto de la imagen (2.25) y (2.26). Resolver estas ecuaciones simultáneas por algún método estándar como el de Gauss Jordan (Lay, 2001) resultaría muy costoso ya que el número de columnas y renglones de la matriz corresponde a la dimensión de la imagen en pixeles. Por lo tanto se aplica el método iterativo de Gauss Seidel (Lay, 2001) para su solución dado por las siguientes fórmulas:

$$u_{x,y}^{k+1} = u_{x,y}^{-k} - \frac{I_x\left[I_x u_{x,y}^{-k} + I_y v_{x,y}^{-k} + I_t\right]}{\alpha^2 + I_x^2 + I_y^2} \tag{2.27}$$

$$v_{x,y}^{k+1} = v_{x,y}^{-k} - \frac{I_y\left[I_x u_{x,y}^{-k} + I_y v_{x,y}^{-k} + I_t\right]}{\alpha^2 + I_x^2 + I_y^2} \tag{2.28}$$

donde $\left[u_{x,y}^{k}, v_{x,y}^{k}\right]$ son la estimación de velocidad para el píxel (x, y) y $\left[u_{x,y}^{-k}, v_{x,y}^{-k}\right]$ son el promedio de los vecinos de $\left[u_{x,y}^{k}, v_{x,y}^{k}\right]$. Para $k = 0$, la velocidad inicial es 0. La estimación de la velocidad en un punto, no depende de la estimación previa de velocidad del mismo punto.

Algoritmo de Horn-Schunck para el cálculo de flujo óptico:

1. Calcular I_x e I_y utilizando el filtro de convolución de Sobel $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$ y su traspuesta $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ para cada pixel en la primera imagen.
2. Calcular I_t entre las imágenes uno y dos utilizando la máscara $\begin{bmatrix} -1 & 1 \end{bmatrix}$.

3. Asumir que la velocidad inicial es 0 y calcular la velocidad promedio para cada píxel utilizando $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ como filtro de convolución.
4. Iterativamente resolver para u y v aplicando las ecuaciones (2.27) y (2.28).

$$u_{x,y}^{k+1} = u_{x,y}^{-k} - \frac{I_x\left[I_x u_{x,y}^{-k} + I_y v_{x,y}^{-k} + I_t\right]}{\alpha^2 + I_x^2 + I_y^2} \tag{2.27}$$

$$v_{x,y}^{k+1} = v_{x,y}^{-k} - \frac{I_y\left[I_x u_{x,y}^{-k} + I_y v_{x,y}^{-k} + I_t\right]}{\alpha^2 + I_x^2 + I_y^2} \tag{2.28}$$

Características del algoritmo de Horn y Schunck:

- Supone un brillo constante y las velocidades de los píxeles del entorno prácticamente idénticas. Incorpora información global.
- Únicamente usa las primeras derivadas de la imagen.
- Utiliza un método iterativo. Supone también un movimiento suave a través de los bordes.

2.5.3.2 Algoritmo de Lucas-Kanade

Las técnicas de registro tienen una gran variedad de aplicaciones en visión por computadora como la visión estéreo, reconocimiento de patrones y análisis de movimiento. Lucas Bruce y Takeo Kanade (1981) presentan una técnica de registro de imágenes que utiliza los gradientes de intensidad espacial para la búsqueda directa de posiciones de coincidencias entre imágenes (localización de un punto de una imagen a otra). Esta técnica fue desarrollada para aplicaciones estéreo, pero resultó aplicable para cualquier número de dimensiones. La ecuación de error propuesta por Lucas y Kanade es la siguiente:

$$\varepsilon = \sum_{x \in R} [F(x+h) - G(x)]^2 \tag{2.29}$$

Aquí x y h son n dimensiones de vectores.

Los investigadores Barron, Fleet y Bauchemin (1992,1995) aplicaron las técnicas de minimización de error de Lucas y Kanade, a la ecuación restringida para el cálculo de flujo óptico (2.8) de Horn y Schunck (1981), por medio de la ponderación de mínimos cuadrados. Desde entonces se le conoce a esta aplicación como algoritmo de Lucas-Kanade para el cálculo del flujo óptico. Este algoritmo permite la extracción de las dos componentes de la velocidad utilizando una función ventana para tomar los valores de los gradientes espaciales y temporales de un entorno de vecindad de un punto. Propone una estimación del flujo óptico por bloques, es decir, determina el vector de movimiento para un bloque de píxeles. Esto significa dividir la imagen original en pequeñas secciones y suponer una velocidad constante en cada sección. Este método, adapta la ecuación restringida de cálculo de flujo óptico a un modelo de vecindad constante $[u \quad v]^T$ en cada sección Ω, reduciendo al mínimo la siguiente función de error:

$$\sum_{p_i \in \Omega} W^2(x,y) \left[I_x u + I_y v + I_t \right]^2 \tag{2.30}$$

Aquí, $W(x,y)$ es la función ventana que acentúa las restricciones en el centro de cada sección Ω.

La solución al problema de la minimización es dada por la siguiente ecuación:

$$A^T W^2 A v = A^T W^2 b\,, \tag{2.31}$$

donde para cada pixel $p_i \in \Omega$ en el instante t:

A es la matriz de vectores de un cuadro de imagen,

$$A = \begin{bmatrix} I_x(p_1) \ldots I_x(p_n) \\ I_y(p_1) \ldots I_y(p_n) \end{bmatrix}^T \tag{2.32}$$

W es la diagonal que contiene los valores de restricción,

$$W = diag[W(p_1)...W(p_n)]^T \tag{2.33}$$

b representa las posiciones en el siguiente cuadro de la imagen,

$$b = -[I_t(p_1)...I_t(p_n)]^T \tag{2.34}$$

Resolviendo por mínimos cuadrados,

$$\begin{bmatrix} \sum W^2 I_x^2 & \sum W^2 I_x I_y \\ \sum W^2 I_y I_x & \sum W^2 I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} \sum W^2 I_x I_t \\ \sum W^2 I_y I_t \end{bmatrix} \tag{2.35}$$

Algoritmo de Lucas-Kanade para el cálculo de flujo óptico:

1. Calcular I_x e I_y utilizando el filtro $\frac{1}{12}[-1 \quad 8 \quad 0 \quad -8 \quad 1]$ y su traspuesta.
2. Calcular I_t entre las imágenes uno y dos utilizando la máscara $[-1 \quad 1]$.
3. Suavizar los gradientes I_x, I_y e I_t utilizando un separador y un filtro isotrópico de tamaño $5x5$ elementos cuyos coeficientes son $\frac{1}{16}[1 \quad 4 \quad 6 \quad 4 \quad 1]$.
4. Resolver el sistema de dos ecuaciones lineales con dos incognitas, para cada píxel utilizando el siguiente método:

Si $$A = \begin{bmatrix} a & b \\ b & c \end{bmatrix} = \begin{bmatrix} \sum W^2 I_x^2 & \sum W^2 I_x I_y \\ \sum W^2 I_y I_x & \sum W^2 I_y^2 \end{bmatrix} \tag{2.36}$$

Entonces los valores propios de A son,

$$\lambda_i = \frac{a+b}{2} \pm \frac{\sqrt{4b^2 + (a-c)}^2}{2}, i = 1,2. \tag{2.37}$$

Aquí, λ_i equivale al error cuadrático o residuo.

Cuando el bloque encuentra los valores propios (errores cuadráticos), los compara con el umbral τ, el cual corresponde al valor que se selecciona como umbral

para la reducción del nivel de ruidos. Los resultados están en tres de los siguientes casos:

Caso 1: $\lambda_1 \geq \tau$ y $\lambda_2 \geq \tau$. A es no singular, así que el bloque soluciona el sistema de ecuaciones usando regla de Cramer (Lay, 2001).

Caso 2: $\lambda_1 \geq \tau$ y $\lambda_2 < \tau$. A es singular, así que el bloque normaliza el flujo del gradiente para calcular u y v.

Caso 3: $\lambda_1 < \tau$ y $\lambda_2 < \tau$. El flujo óptico u y v es 0.

Se utiliza el parámetro del umbral para la reducción del nivel de ruidos y eliminar el efecto de movimientos pequeños en cada imagen. Cuanto más alto es el número, menos afectan los movimientos pequeños al cálculo de flujo óptico.

Características del algoritmo de Lucas-Kanade:

- La velocidad se supone localmente constante.
- Su solución es de forma cerrada cuando A es no singular.
- Utiliza un umbral que contribuye a la reducción del nivel de ruidos.

2.5.3.3 Algoritmo Piramidal de Lucas-Kanade o multirresolución

Este algoritmo (Bouquet, 2000) implementa una solución iterativa aplicando pirámides Gaussianas (Nuño, Arias y Feregrino, 2005) al algoritmo de Lucas-Kanade (1981). Calcula las coordenadas de un punto en un cuadro de la secuencia, dadas las coordenadas en el cuadro anterior. El procedimiento consiste en calcular una imagen que en cada píxel contiene la media (u otra distribución, gausiana, laplaciana.) de él con sus vecinos (por ejemplo los ocho más próximos) para a continuación submuestrear la imagen a la mitad. De esta manera se obtiene la representación de la imagen en varios niveles jerárquicos. El primer nivel sería la imagen original, el siguiente nivel sería la imagen emborronada y submuestrada, y así sucesivamente. La Figura 2.9, ilustra las Pirámides Gaussianas entre dos cuadros en una secuencia de imágenes. La Figura 2.10,

ilustra la aplicación del algoritmo de Lucas-Kanade en cada nivel piramidal correspondiente a dos imágenes.

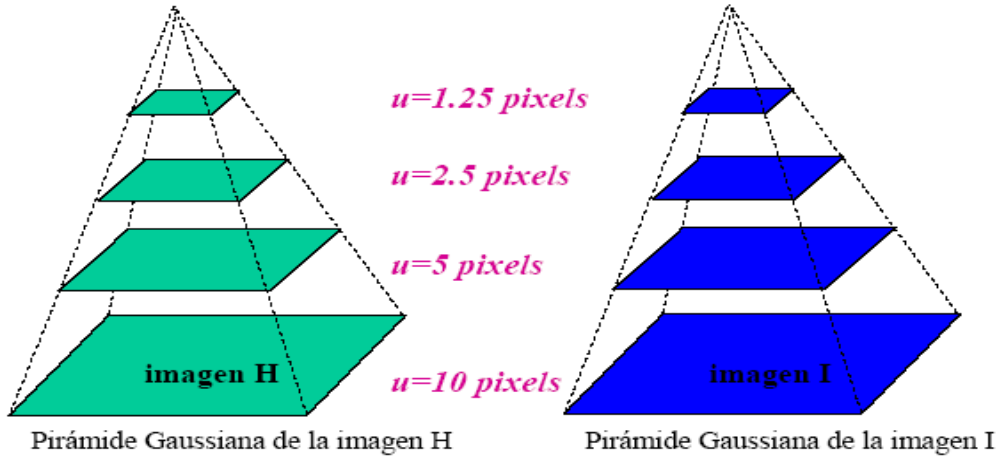


Figura 2.9. Pirámides Gaussianas entre dos cuadros en una secuencia de imágenes.

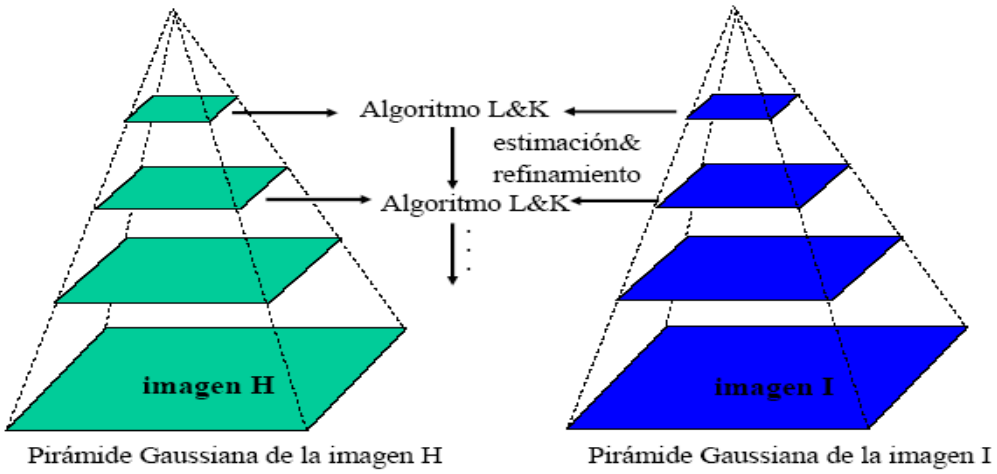


Figura 2.10 Aplicación de Lucas Kanade con Pirámides Gaussianas.

Algoritmo iterativo piramidal de Lucas-Kanade para el cálculo de flujo óptico.

1. Obtener una representación piramidal (N niveles) a través de dos imágenes consecutivas de la secuencia. Sea N el nivel más borroso y 0 el nivel inicial (imagen original).
2. Desde $nivel = N$ decrementa hasta 0 en pasos de -1:

a) Calcular el vector velocidad del punto seleccionado por medio del flujo óptico con método de Lucas y Kanade. La velocidad de un punto dará la posición del punto en la imagen de la secuencia posterior.
b) Un punto del $nivel\ i$, corresponde a un bloque de puntos del $nivel\ i-1$. Si las velocidades de los puntos del $nivel\ i$ (son calculadas en el punto anterior), se obtiene la velocidad de un bloque en el $nivel\ i-1$. Repetir: $nivel = nivel - 1$ para conseguir la posición del punto con la resolución de la imagen original.

Características del algoritmo piramidal de Lucas-Kanade:
Mejora el comportamiento del filtro en áreas con gran movimiento.

2.5.4 Aplicación de restricciones al flujo óptico obtenido

La última fase del procedimiento para el cálculo de flujo óptico consiste en aplicar restricciones para minimizar las posibles fallas en la ecuación de cálculo de flujo óptico. Algunas restricciones son implícitas como funciones de error en cada algoritmo. Por lo tanto, se considera al cálculo del flujo óptico y la de aplicación de restricciones como una sola fase. Existen otras restricciones que se aplican para seleccionar los vectores de flujo óptico adecuados a la secuencia de imágenes analizada. Estas restricciones consisten en eliminar aquellos valores de flujo óptico que no cumplan con un rango establecido o aquellos que no pertenezcan a áreas seleccionadas como bordes o esquinas de los objetos en la escena.

2.6 Descripción de las plataformas de desarrollo utilizadas en la implementación de los algoritmos para el cálculo de flujo óptico.

En esta sección, se describen las plataformas de desarrollo utilizadas en la implementación de los algoritmos para el cálculo de flujo óptico de acuerdo al método de gradientes.

2.6.1 Plataforma MATLAB versión 6.1

MATLAB es una herramienta de software de alto rendimiento para cálculo (Nakamura, 1997; The MathWorks, Inc., 2004-2007). Integra el cálculo matemático, la visualización

de resultados y la programación de rutinas en un ambiente fácil de utilizar, puesto que permite la escritura de ecuaciones en notación matemática conocida. La biblioteca matemática de MATLAB facilita los análisis matemáticos de tal modo que el usuario crea rutinas matemáticas adicionales con mayor facilidad que en otros lenguajes de programación, debido a la continuidad entre las variables, puesto que aquí no hay distinción entre los números reales, complejos, enteros de precisión simple o de doble precisión puesto que las variables no requieren de una declaración previa durante la programación. Por lo tanto, esta herramienta facilita la creación rápida de programas para el análisis y pruebas. Las aplicaciones típicas incluyen: matemáticas y cálculo, desarrollo de algoritmos, adquisición de datos, modelado, simulación y prototipos, análisis de datos, exploración y visualización de los mismos, gráficos científicos y de ingeniería y por supuesto, el desarrollo de aplicaciones.

El nombre MATLAB significa laboratorio de matrices, fue escrito originalmente para proporcionar el acceso fácil al software para matrices desarrollado por los proyectos de LINPACK y de EISPACK (The MathWorks, Inc., 2004-2007). Hoy, MATLAB incorpora las bibliotecas de LAPACK y de BLAS. Es por ello que MATLAB fue de utilidad en el desarrollo de este proyecto en el cual se analizan imágenes y siendo una imagen una matriz de datos, esta plataforma, proporciona las herramientas adecuadas para tal objetivo. La utilización de un lenguaje de programación de propósito general sin alguna herramienta adicional que contenga una biblioteca de funciones para cálculos con matrices, significa un alto costo de tiempo para la implementación de algoritmos y desarrollo de proyectos, ya que se requeriría escribir cada función de cálculo, además de funciones que resuelvan cada paso de los algoritmos para el procesamiento de imágenes.

Con esta herramienta se diseñaron programas de prueba entre dos cuadros de imagen, con la finalidad de analizar los algoritmos de cálculo de flujo óptico por gradientes, seleccionados para la implementación. Estos programas de prueba, permitieron observar resultados de los datos de salida con respecto a los parámetros de entrada. Este análisis se describe en la sección de análisis y prueba.

2.6.2 Plataforma MATLAB versión 2006a

Otra de las plataformas utilizadas fue MATLAB versión 2006a y 2006b que contiene una herramienta de procesamiento de imágenes y video incluida en un ambiente de simulación Simulink. Esta herramienta es conocida como *"Video and Image Processing Blockset"* (The MathWorks, Inc., 2004-2007) y se utiliza para el diseño rápido de prototipos, la simulación gráfica y la generación eficiente del código de algoritmos de procesamiento de video por medio de bloques. Los bloques de este procesador son para capturar video, filtrar imágenes, realizar la valoración del movimiento, detectar bordes y otros algoritmos de procesamiento de señal. Algunas de las ventajas que proporciona esta plataforma son:

- Permite ejecutar el diseño del sistema en tiempo real.
- La observación gráfica de los bloques de cada función.
- La variación de los parámetros de entrada y verificación de los parámetros de salida correspondientes.
- Diseño rápido del prototipo con los componentes principales de un sistema.
- Permite un análisis visual del problema facilitando su comprensión y posibles soluciones.

Por otro lado, los inconvenientes de esta herramienta son que los bloques de funciones están limitados a determinados parámetros de entrada y de salida y el rango de valores también está limitado. Otro inconveniente es que la velocidad de cómputo con MATLAB es significativamente más baja que con Lenguaje "C" u otros lenguajes de programación, ya que el precio que se paga por crear un ambiente fácil de trabajo y utilizar gráficos es el consumo de memoria y mayor tiempo de ejecución en los procesos.

2.6.3 Plataforma OpenCV

La biblioteca OpenCV *(Open Source Computer Vision Library)*, es una colección de funciones de C y algunas clases de C++ que implementan algoritmos conocidos para el procesamiento de imágenes y de visión por computadora (OpenCV, 2000-2006). Esta colección de funciones pertenece a la compañía Intel y proporciona una plataforma de

nivel medio-a-alto que incluye cerca de 300 funciones de C y algunas clases de C++. OpenCV es un API que está mejorando constantemente gracias a que es de uso libre *(Open Source)* y no tiene ninguna dependencia estricta con bibliotecas externas, aunque utiliza bibliotecas adicionales para procesamiento de imágenes escritas en "lenguaje C" que son también de libre distribución o restringidas por licencias. Ejemplos de bibliotecas que se le añaden son las siguientes:

- libjpeg que contiene funciones para compresión y descompresión de imágenes.
- ffmpeg que es una colección de software para captura y conversión de audio y video.
- GTK que permite crear interfaces gráficas de usuarios de fácil uso.

Estas tres bibliotecas son creadas por fundaciones de libre distribución, también emplea la biblioteca *"Image Processing Library"* (IPL) de Intel que implementa funciones de bajo nivel para el procesamiento de imágenes, pero requiere licencia para su utilización. Se utiliza OpenCV para la implementación final porque proporciona un marco de trabajo de alto nivel para el desarrollo de aplicaciones de visión por computadora y por su optimización para funcionar en tiempo real y en diversas plataformas, logrando con esto un software transportable.

CAPÍTULO III

MODELADO DEL SISTEMA PARA CÁLCULO DE FLUJO ÓPTICO

El objetivo general de este proyecto es desarrollar soluciones de software que permitan calcular el flujo óptico a partir de una secuencia de imágenes capturada en tiempo real por una cámara monocular. El sistema debe obtener información del movimiento aparente entre un cuadro y otro, construyendo así un sistema por computadora, el cual sirva como módulo de visión para aplicaciones específicas que requieren procesar parámetros simples de la escena tales como, la dirección y magnitud del movimiento. Para el cumplimiento de este objetivo, se realizan diversas tareas llevadas a cabo en tres etapas. Dos de estas etapas se exponen en este capítulo.

En la primera etapa se desarrollan programas de prueba, donde se implementan las diferentes técnicas y métodos de análisis de escenas dinámicas basados en gradientes vistas en el capítulo II. La finalidad de estos programas es detectar los cambios entre dos cuadros sucesivos de una secuencia de imágenes y determinar la viabilidad de los algoritmos de Horn-Schunck y Lucas-Kanade, en cuanto al procesamiento de los datos, y detección de vectores de flujo óptico que indiquen la magnitud y sentido del movimiento presente en la secuencia. En esta etapa se utiliza como plataforma MATLAB versión 6.1 la cual cuenta con funciones básicas de procesamiento de imágenes y permite el cálculo de operaciones matemáticas con matrices.

En la segunda etapa, se diseña un modelo aproximado para detección de movimiento en tiempo real, utilizando módulos de procesamiento de imágenes y captura de video, mediante la herramienta *"Video and Image Processing Blockset"* la cual es parte del software MATLAB versión 2006a. Este software permite estructurar sistemas por medio de diagramas de bloques que dividen el programa en etapas donde es posible modificar los valores de los parámetros de entrada para el análisis de los resultados, logrando flexibilidad en el procesamiento.

En la última etapa, se desarrolla el Sistema para Cálculo de Flujo Óptico, empleando la biblioteca de funciones conocida como OpenCV de la compañía Intel de libre distribución, la cual facilita el diseño de programas de procesamiento de imágenes funcionales en diversas plataformas debido a que el lenguaje de programación utilizado es "C". Además brinda la posibilidad de modificar el código de las funciones contenidas en la biblioteca, permitiendo su adaptación a tareas específicas y nuevas aportaciones a los algoritmos. Dicha biblioteca fue creada para el procesamiento de imágenes en tiempo real ya que ejecuta operaciones a nivel procesador, es decir, sus funciones ingresan información en formato binario, la cual es ejecutada directamente por el procesador sin conversión alguna. Esta etapa es expuesta en el capítulo IV.

3.1 Análisis y prueba de los algoritmos para cálculo de flujo óptico

En esta etapa se desarrollan programas prueba donde se implementan algoritmos para cálculo de flujo óptico basados en el método de gradientes espacio-temporales. Estos algoritmos fueron seleccionados de acuerdo a la investigación documental realizada, misma que arroja como consecuencia, la existencia de dos algoritmos básicos, el de Horn-Schunck (1981) y el de Lucas-Kanade (1981), y existen a la fecha, diversas variaciones de los mismos.

Para la realización de las pruebas, se utiliza MATLAB versión 6.1 como herramienta de programación, ya que es un software diseñado principalmente para cálculos con matrices y cuenta con algunas funciones de procesamiento de imágenes. Estas pruebas se llevan a cabo examinando dos cuadros sucesivos en una escena dinámica, ya que esta es la cantidad mínima requerida para ser considerada como una secuencia de imágenes. Cada cuadro representa un instante de tiempo (t_0, t_1). Las imágenes que aparecen en dichos cuadros son imágenes sintéticas para facilitar la evaluación de los métodos, no se utiliza sensor de imagen alguno.

3.1.1 Desarrollo de programas de prueba aplicando el algoritmo de Horn-Schunck

En los programas prueba donde se implementa el algoritmo de Horn-Schunck, se agrega la técnica de umbralización, para obtener los bordes de los objetos móviles y reducir la

cantidad de información a procesar. La razón de agregar estás técnicas es porque el algoritmo de Horn-Schunck calcula los parámetros de flujo óptico punto por punto, lo que implica una alta exigencia en el uso de recursos computacionales.

3.1.1.1 Algoritmo

1. Calcular I_x e I_y utilizando el filtro de convolución de Sobel $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$ y su traspuesta $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ para cada pixel en la primera imagen.
2. Calcular I_t entre las imágenes uno y dos utilizando la máscara $\begin{bmatrix} -1 & 1 \end{bmatrix}$.
3. Asumir que la velocidad inicial es 0 y calcular la velocidad promedio $\bar{v}$ para cada píxel utilizando $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ como filtro de convolución.
4. Resolver iterativamente para u y v según las fórmulas 2.27 y 2.28 del capítulo II .

$$u_{x,y}^{k+1} = u_{x,y}^{-k} - \frac{I_x\left[I_x u_{x,y}^{-k} + I_y v_{x,y}^{-k} + I_t\right]}{\alpha^2 + I_x^2 + I_y^2} \tag{2.27}$$

$$v_{x,y}^{k+1} = v_{x,y}^{-k} - \frac{I_y\left[I_x u_{x,y}^{-k} + I_y v_{x,y}^{-k} + I_t\right]}{\alpha^2 + I_x^2 + I_y^2} \tag{2.28}$$

Aquí, $\begin{bmatrix} u_{x,y}^k & v_{x,y}^k \end{bmatrix}$ es la velocidad estimada para el pixel (x,y) y $\begin{bmatrix} u_{x,y}^{-k} & v_{x,y}^{-k} \end{bmatrix}$ es el promedio de la vecindad de $\begin{bmatrix} u_{x,y}^k & v_{x,y}^k \end{bmatrix}$. Para $k = 0$, la velocidad inicial es 0.

3.1.1.2 Implementación del programa de prueba

1. Se lee la imagen de cada cuadro.
2. Se convierte cada imagen a niveles de gris ya que el flujo óptico se entiende por el cambio de los niveles de gris de un cuadro a otro, debido al movimiento de los objetos.

3. Se aplica el filtro de Sobel $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$ y su traspuesta a cada cuadro para obtener los gradientes espaciales I_x e I_y (Este paso corresponde al paso 1 de algoritmo general de Horn-Schunck). En este paso se obtienen 4 imágenes, las cuales son los gradientes espaciales I_x y I_y para t_0 y t_1.
4. El resultado del filtrado del primer cuadro se convierte a una matriz de niveles de gris para manipular la información en forma numérica.
5. Se calcula la diferencia (filtro $[-1 \quad 1]$) entre el primero y el segundo cuadro tomando los valores absolutos de los vectores, ya que el valor requerido es la magnitud, obteniendo de este modo el gradiente temporal. (Este paso es equivalente al paso 2 de algoritmo general de Horn-Schunck).
6. El gradiente temporal se multiplica por el gradiente espacial para obtener la imagen resultante (imagen inicial más la diferencia).
7. Se normalizan los valores en el rango de 0 a 255 (escala de grises).
8. La matriz resultante de valores de gris, es una imagen que tiene el mismo tamaño que la imagen inicial.
9. En este último paso se aplica una técnica para distinguir fondo de objetos o características de interés en la imagen. En este caso se obtienen los bordes de los objetos, los cuales, en ocasiones resultan de menor valor que el fondo, por lo que se utiliza la umbralización para distinguirlos. De esta forma se obtiene la información de utilidad. Se eligen dos umbrales T_1 y T_2, tales que todos los puntos que se ubican dentro del rango $T_1 \leq f(x,y) \leq T_2$ son los valores de los objetos en la escena y los que están fuera $f(x,y) > T_2$ ó $f(x,y) < T_1$ corresponden al fondo.

La ecuación siguiente (3.1) define la umbralización para el despliegue de la imagen,

$$g(x,y) = \begin{cases} 1, & si \quad T_1 \leq f(x,y) \leq T_2 \\ 0, & si \quad f(x,y) > T_2 \text{ ó } \quad f(x,y) < T_1 \end{cases} \tag{3.1}$$

El paso número 9 es una modificación para la comprobación del algoritmo de Horn-Schunck que permite la reducción de cálculos y agiliza el proceso. Debido a que

se incluyó imágenes sintéticas de prueba, donde la escena consta de un objeto que se mueve solo entre dos cuadros de escena, se aplica la técnica de umbrales para distinguir el objeto del fondo. Con esta aplicación, se reduce el cálculo de gradientes solamente a los bordes del objeto para una ejecución más rápida, ya que la plataforma MATLAB 6.1 para este tipo de análisis, a mayor número de cálculos, el proceso resulta más lento (ver Apéndice A). El diagrama de flujo correspondiente a este algoritmo, se verifica en la Figura (3.1).

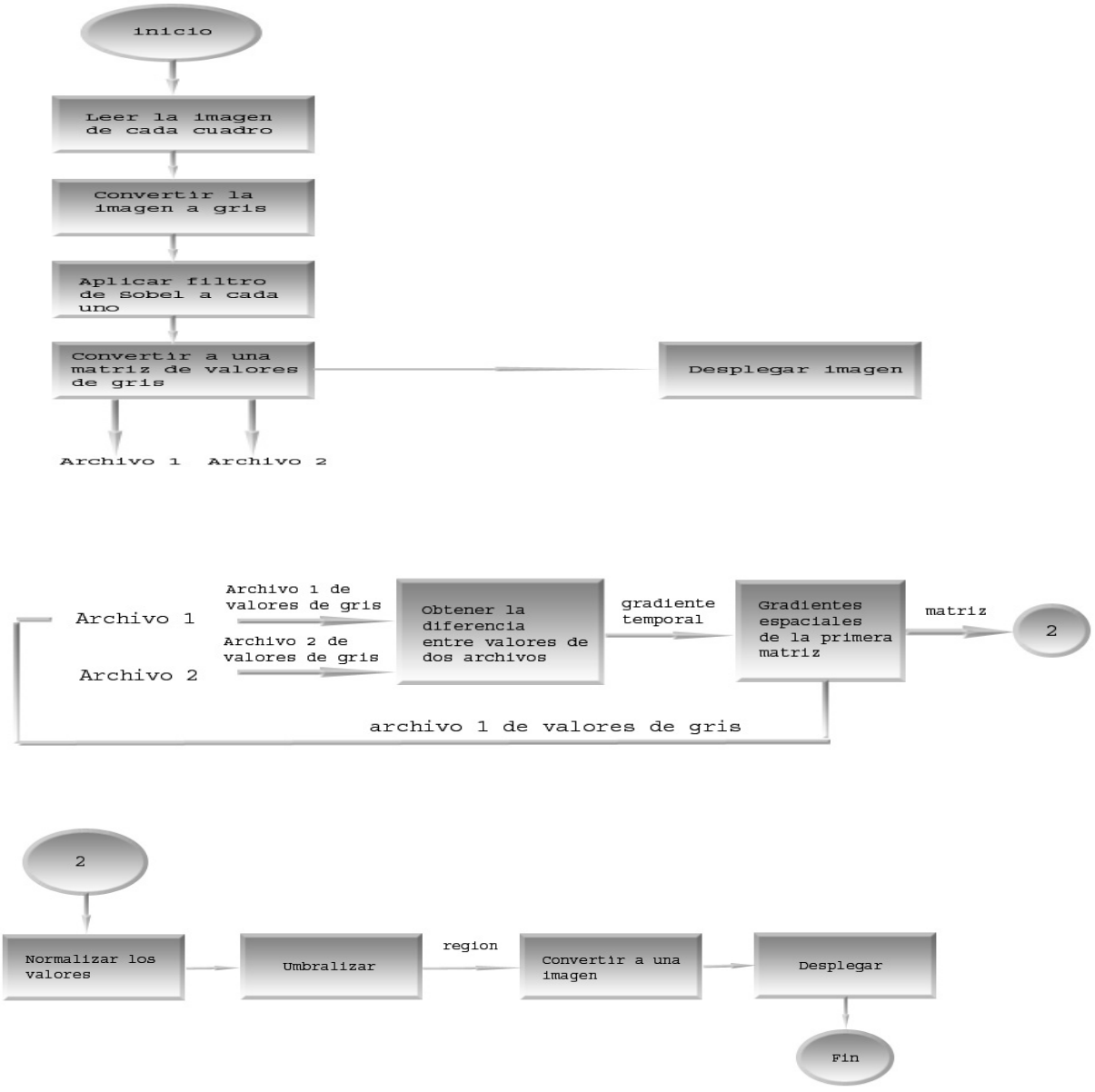


Figura 3.1. Diagrama de flujo correspondiente al programa de prueba implementado para el algoritmo de Horn-Schunck.

3.1.1.3 Comprobación de movimiento mediante gradientes horizontales

A continuación se muestran de manera gráfica los resultados obtenidos de la implementación del algoritmo de Horn-Schunck. Para la obtención de los resultados, se aplican como parámetros de entrada imágenes sintéticas que contienen un objeto y un fondo. Se analizan primeramente dos cuadros de imagen donde el objeto aparece en distinta posición en cada cuadro, con la finalidad de comprobar la detección de movimiento (ver Figuras 3.2, 3.3, 3.4). Posteriormente se analizan dos cuadros en donde el objeto se encuentra en la misma posición en instantes diferentes de tiempo, siendo el objetivo de este análisis la comprobación de ausencia de movimiento (ver Figuras 3.5, 3.6). Ambos casos deben de cumplir con la ecuación para el cálculo de flujo óptico referida en el capítulo II.

$$I_x V_x + I_y V_y + I_t = 0 \tag{2.8}$$

Los resultados mostrados en la siguiente imagen (ver Figura 3.2) se obtienen calculando el gradiente horizontal (I_x) de cada imagen y el correspondiente gradiente temporal. Utilizando un solo gradiente espacial se logró determinar el movimiento del objeto en la imagen.

En la Figura 3.2, se observa el resultado de los procesos aplicados a cada cuadro. Las gráficas a) y b) corresponden a las imágenes capturadas X y Y. Las gráficas c) y d) corresponden a los gradientes horizontales de cada una de las imágenes, mostradas en niveles de gris. Los bordes destacan en la componente horizontal. Por último, se observa en la gráfica e) el cambio ocurrido entre un cuadro y otro por medio del cálculo del gradiente temporal y espacial. La multiplicación del gradiente temporal obtenido de la diferencia entre ambas imágenes, por el gradiente espacial de la primera imagen, permite visualizar cada objeto en distinta posición y su desplazamiento en una misma gráfica (ver Apéndice A).

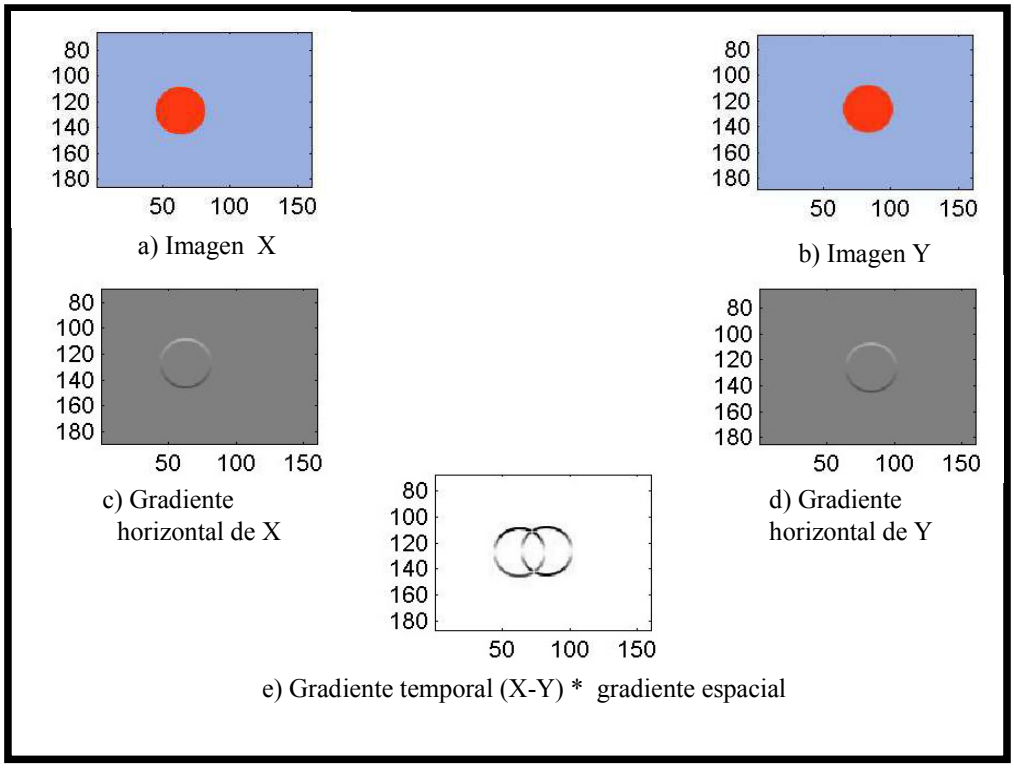


Figura 3.2. Gráficas del movimiento de un objeto en instantes distintos de tiempo.

3.1.1.4 Comprobación de movimiento con ambos gradientes espaciales (horizontal y vertical) y la representación vectorial de los gradientes temporales

Este programa está basado en el mismo proceso de implementación del algoritmo de Horn-Schunck para dos cuadros de imagen, con la diferencia de que los gradientes espaciales son obtenidos en forma independiente para representar los gradientes temporales de modo vectorial (ver Apéndice A).

En la Figura 3.3, las gráficas a) y b) muestran las imágenes originales de cada cuadro. En c) y d) se grafican los gradientes horizontales de ambas imágenes y en e) y f) se grafican los gradientes verticales de las mismas

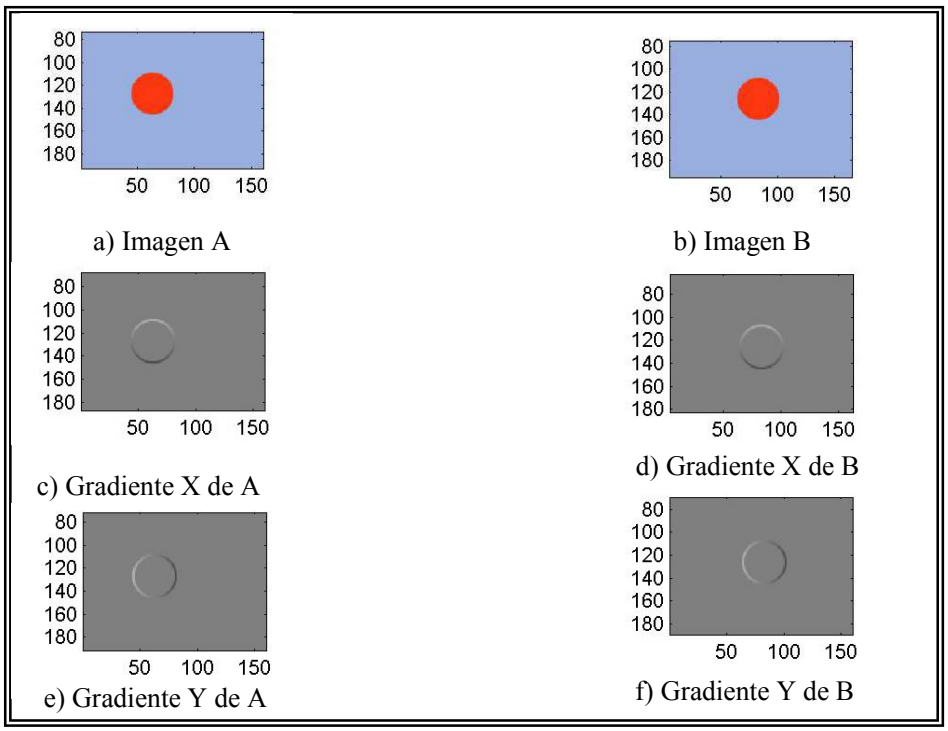


Figura 3.3. Cálculo de ambos gradientes espaciales (horizontal y vertical).

En la Figura 3.4, se observa de modo vectorial la representación de los gradientes temporales que determinan el movimiento entre los dos cuadros de imagen. También se observa que la orientación de los vectores es en sentido contrario en cada objeto de la imagen. Por lo tanto, la suma de los gradientes temporales resultaría cero tal y como lo expresa la ecuación restringida de cálculo de flujo óptico descrita en el capítulo II.

$$I_x V_x + I_y V_y + I_t = 0 \tag{2.8}$$

Esta representación gráfica vectorial se obtiene mediante la función *quiver*( ) de MATLAB. La función *quiver*( ), despliega los vectores de velocidad en forma de flechas utilizando los componentes (u, v) en los puntos (x, y) . Por ejemplo, $quiver(x, y, u, v)$ traza los vectores en forma de flechas en las coordenadas especificadas por cada par de elementos en x y y. Las matrices x, y, u, y v deben ser del mismo tamaño y contener componentes correspondientes a la posición y a la velocidad. Sin embargo, x y y pueden también ser vectores.

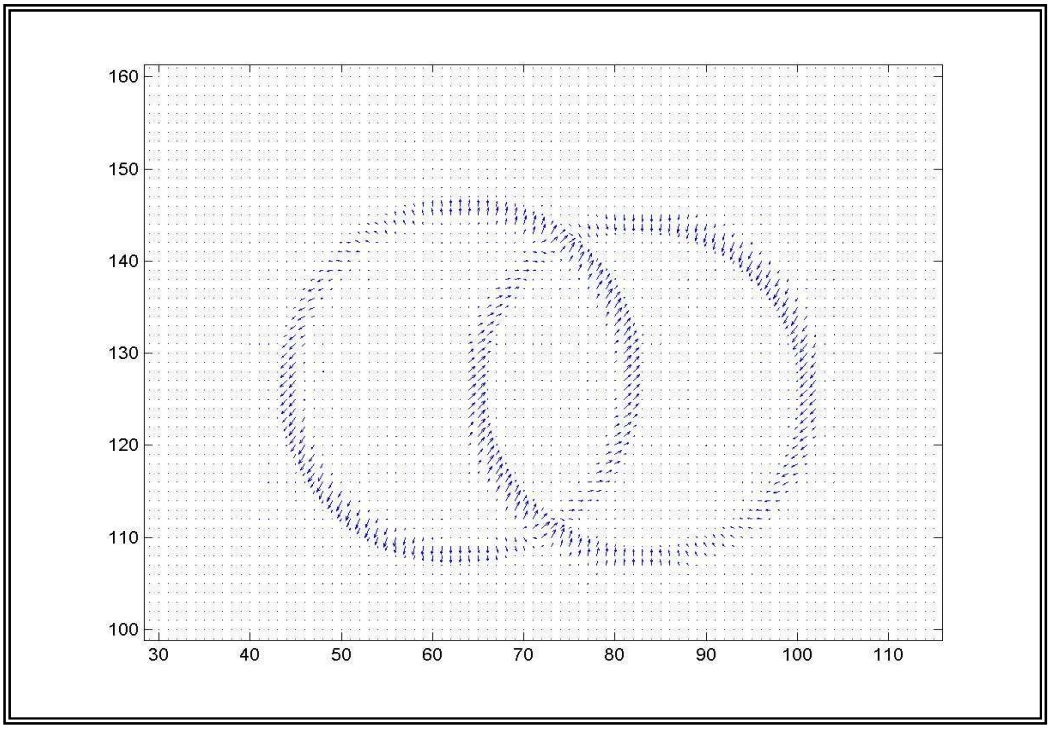


Figura 3.4. Representación vectorial de gradientes temporales.

3.1.1.5 Comprobación de la ausencia de movimiento entre dos cuadros de una escena

En la Figura 3.4, se observa que la orientación en los vectores de flujo en la primera imagen es en sentido contrario a la segunda imagen. Por lo tanto, se concluye que si calculamos los gradientes temporales en dos cuadros de imagen donde el objeto se encuentra en la misma posición, la representación vectorial estará vacía, debido a que no se grafica un vector con valor de cero, el cual es el resultado de los gradientes temporales cuando no hay movimiento. Para comprobar si el método es capaz de determinar la ausencia de movimiento bajo este criterio, se utiliza dos veces la misma imagen efectuando los procesos descritos anteriormente para calcular la diferencia de gradientes, obteniendo los siguientes resultados (ver Apéndice A). En la Figura 3.5, en a) y b) se muestran las imágenes originales de cada cuadro donde el objeto se observa en la misma posición. En c) y d) se grafican los gradientes horizontales de ambas imágenes y en e) y f) se grafican los gradientes verticales de las mismas.

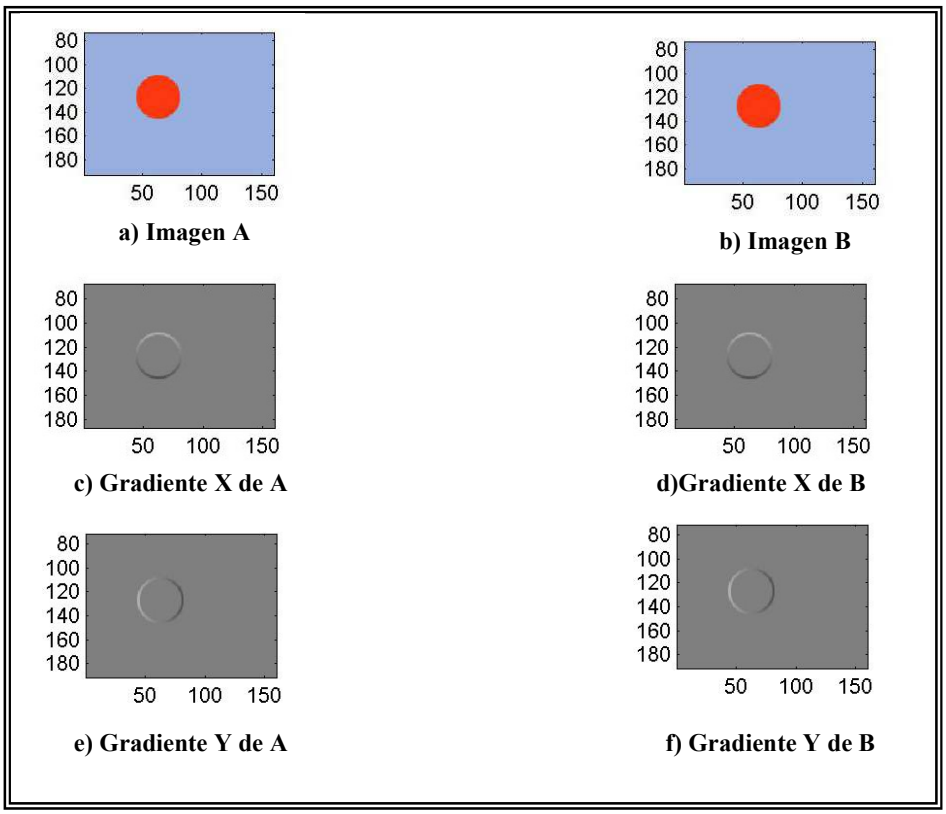


Figura 3.5. Gráficas de resultados correspondientes a dos cuadros de imagen donde el objeto se encuentra en la misma posición.

En la Figura 3.6, se observa la ausencia de vectores debido a que los gradientes temporales de la secuencia de imágenes resultaron con valor de cero. Por lo tanto, se comprueba la ausencia de movimiento entre los dos cuadros de imagen.

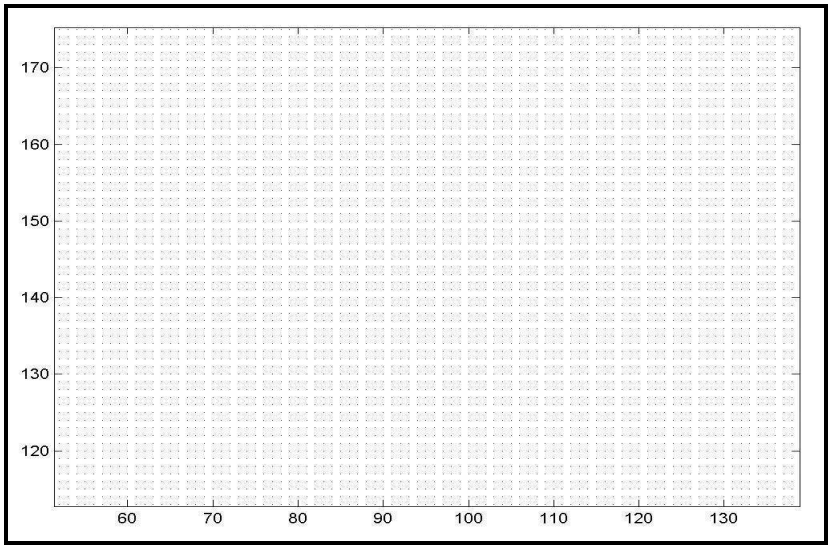


Figura 3.6. Gradiente temporal (X-Y) * espacial = 0

3.1.2 Desarrollo de programas de prueba aplicando el algoritmo de Lucas-Kanade

El algoritmo de Lucas-Kanade difiere del algoritmo de Horn-Schunck en la determinación del vector de movimiento. Lucas-Kanade realiza el cálculo para un bloque de píxeles en lugar de punto a punto. Esto significa dividir la imagen original en pequeñas secciones y suponer una velocidad constante en cada sección, lo que se concluye como una reducción en el proceso de cálculo. La ecuación de error establecida en este algoritmo, vista en el capítulo II es:

$$\sum_{x\in\Omega} W^2(x,y)\ \left[I_x u + I_y v + I_t\right]^2 \tag{2.30}$$

3.1.2.1 Algoritmo

1. Calcular I_x e I_y utilizando el filtro $\frac{1}{12}[-1 \quad 8 \quad 0 \quad -8 \quad 1]$ y su traspuesta en el primer cuadro de imagen.
2. Calcular I_t entre las imágenes uno y dos utilizando el kernel $[-1 \quad 1]$.
3. Suavizar los gradientes I_x, I_y e I_t utilizando un separador y un filtro isotrópico de tamaño $5x5$ elementos cuyos coeficientes son $\frac{1}{16}[1 \quad 4 \quad 6 \quad 4 \quad 1]$.
4. Resolver las ecuaciones lineales de $2x2$ para cada píxel que requiere la siguiente matriz, la cual es expuesta en el capítulo II.

$$\begin{bmatrix} \sum W^2 I_x^2 & \sum W^2 I_x I_y \\ \sum W^2 I_y I_x & \sum W^2 I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} \sum W^2 I_x I_t \\ \sum W^2 I_y I_t \end{bmatrix} \tag{2.35}$$

3.1.2.2 Implementación del programa de prueba

1. Definir el tamaño de las imágenes sintéticas ($20x20$).
2. Crear las matrices correspondientes a las imágenes $A0$ y $A1$.
3. Generar el objeto o región por medio de valores de gris aleatorios, para asignarlo a cada imagen en una posición diferente.

4. Colocar el objeto o región generada, en una posición determinada en cada imagen.
5. Estandarizar los valores de gris de 0 a 255.
6. Fijar el tamaño de los bloques para dividir la región de cálculo del flujo óptico. Esta división se debe a que el algoritmo calcula el flujo óptico por bloques. En este caso particular se utiliza $5x5$.
7. Aplicar el filtro de convolución para el filtrado espacial en ambas imágenes como paso previo al cálculo de gradientes espaciales.

$$I'(x,y) = I(x,y) * h(x,y) = \sum_{i=-\infty}^{i=\infty} \sum_{j=-\infty}^{\infty} I(i,j) \quad h(x-i, y-j) \tag{3.2}$$

8. Calcular los gradientes espaciales Ix e Iy en la primera imagen $A0$ de la siguiente manera:

$$Hx = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}' \tag{3.3}$$

$$Hy = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}, \tag{3,4}$$

donde Hx y Hy corresponden al operador de Prewitt para la extracción de gradientes.

$$Ix = imfilter(A0, Hx, 'simetric') \tag{3.5}$$

$$Iy = imfilter(A0, Hy, 'simetric'). \tag{3.6}$$

La función $imfilter(A,H)$ de MATLAB, extrae los gradientes H de la matriz A.

9. Calcular el gradiente temporal mediante la diferencia de ambas matrices .

$$It = A1 - A0 \tag{3.7}$$

10. Resolver las ecuaciones lineales de $2x2$ para cada píxel utilizando mínimos cuadrados de la ecuación (2.35) expresada anteriormente.
11. Obtener el flujo horizontal y vertical de los bloques de $5x5$ definidos en el paso 6 de este pseudocódigo, para graficar los vectores que representan la magnitud y dirección del movimiento.
12. Se grafican los vectores de flujo óptico mediante la función $quiver(u,v,0)$ de MATLAB.
13. Se grafican las diferencias de movimiento entre la primera y segunda imagen.

En la Figura 3.7, se observan los resultados de la implementación del algoritmo de Lucas-Kanade en una secuencia de dos cuadros de imagen. Las gráficas a) y b) corresponden a las imágenes sintéticas A y B. La gráfica c) muestra la diferencia de posiciones detectada entre A y B. Por último, en la gráfica d) se representan los vectores de dirección y magnitud de flujo óptico para cada bloque en los que se dividen las imágenes.

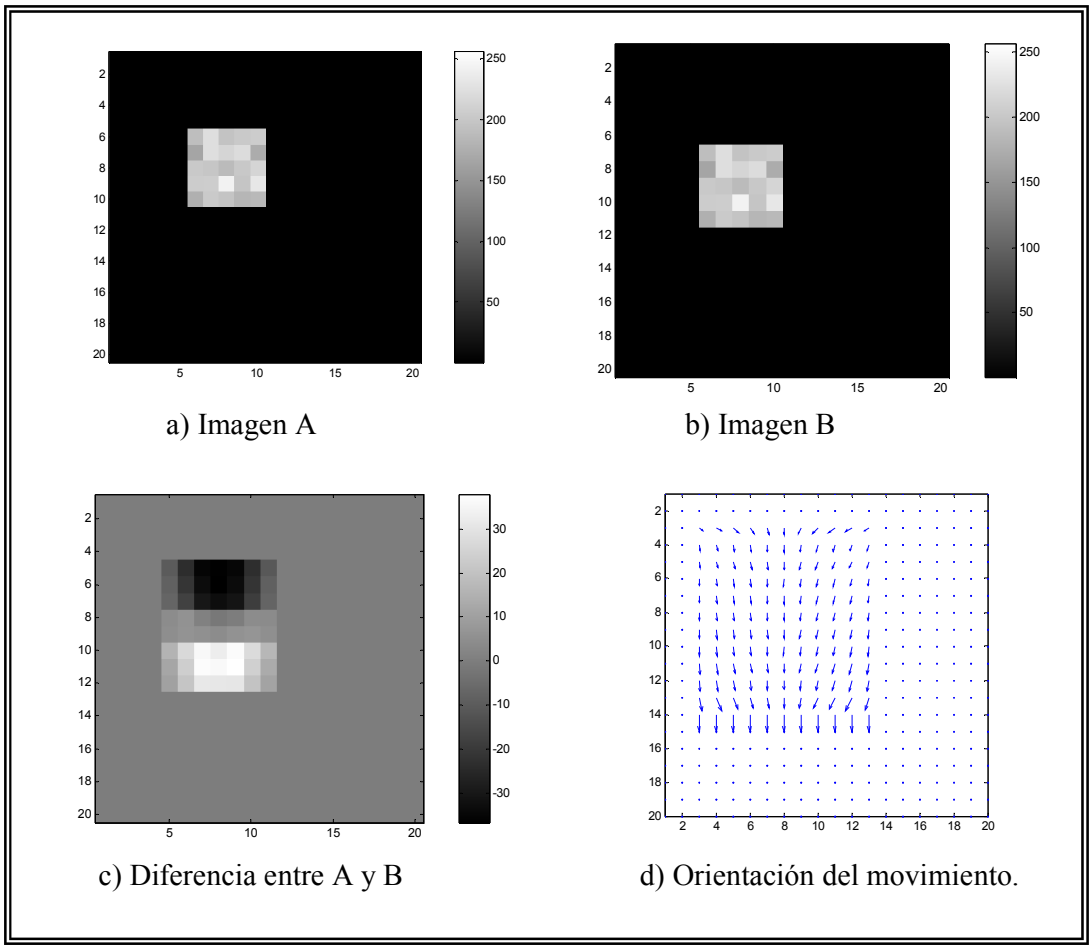


Figura 3.7. Gráficas resultantes de la implementación del algoritmo de Lucas-Kanade con *MATLAB 6.1*.

3.1.3 Conclusiones de la etapa de análisis y prueba

El costo computacional y tiempo de ejecución para el algoritmo de Lucas-Kanade es menor que para el algoritmo de Horn-Shunck. Lucas-Kanade supone un movimiento constante en cada sector en el que se dividió la imagen reduciendo el cálculo del flujo óptico para un píxel promedio en cada bloque. Horn-Schunck calcula el flujo óptico pixel por pixel, lo que implica mayor número de iteraciones para obtener los vectores de

flujo óptico. Para reducir el número de iteraciones y definir la magnitud y dirección de los vectores de flujo óptico al usar el algoritmo de Horn-Schunck, se agregan umbrales para separar el fondo de los objetos de tal modo que se calcula el flujo óptico solo en las áreas convenientes de la imagen.

3.2 Etapa de modelado del sistema para el cálculo de flujo óptico

El objetivo de esta etapa consiste en modelar un sistema capaz de procesar escenas dinámicas simultáneamente a la captura de video, obteniendo los parámetros de movimiento en tiempo real. En base a los resultados obtenidos con las técnicas implementadas en la etapa de análisis y prueba, se diseña un modelo por bloques que representan cada una de los procesos que conforman un sistema de cálculo de flujo óptico para una secuencia de imágenes en tiempo real (ver Figura 3.8). Estos bloques permiten observar de manera gráfica la estructura del sistema, la entrada y salida de datos en cada proceso, además de la manipulación de parámetros en cada uno de ellos. Para lograr este objetivo se utilizó el software Simulink de la plataforma MATLAB versión 2006a, el cual cuenta con la herramienta de modelado y simulación a base de bloques conocida como "*Video and Image Processing Blockset"*. Estos bloques representan funciones de cálculo y procesamiento de matrices, además de funciones de procesamiento de imágenes.

3.2.1 Modelo del sistema

En la Figura 3.8, se muestra el modelo del sistema de cálculo de flujo óptico formado por cinco bloques principales:

- El primer bloque detecta los parámetros del dispositivo de captura de video.
- El segundo bloque convierte la entrada de video R'G'B' a intensidad (niveles de gris).
- El tercer bloque representa la etapa de cálculo de flujo óptico, donde es posible seleccionar entre el algoritmo de Horn-Shunk o el algoritmo de Lucas-Kanade.
- El cuarto bloque corresponde al proceso de umbralización y filtrado de la región de interés (ROI) que se requiere analizar.

- El último bloque despliega los resultados obtenidos del sistema de forma gráfica y numérica en tiempo real.

En las secciones siguientes, se explica brevemente el proceso que realizan cada uno de estos bloques.

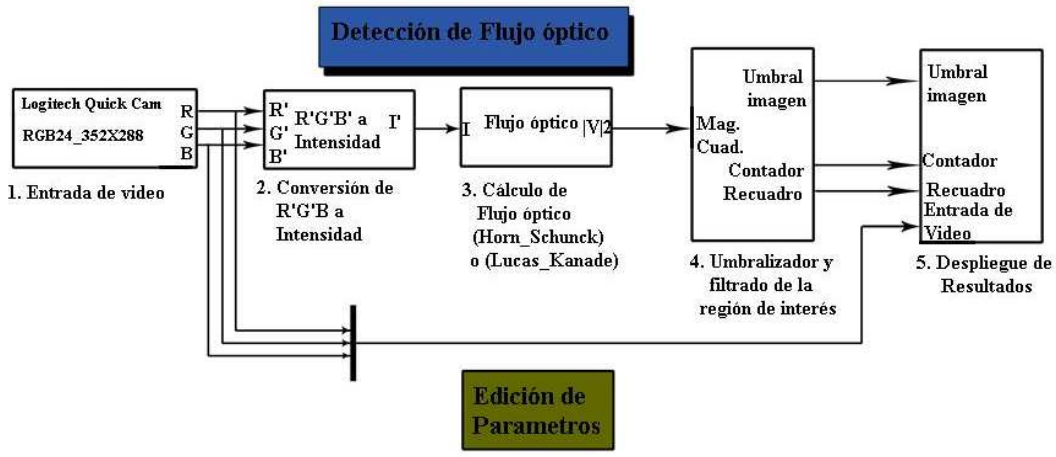


Figura 3.8. Modelo a bloques del sistema de cálculo de flujo óptico en tiempo real.

3.2.2 Bloque de Captura

Este módulo, el cual se observa en la Figura 3.8 (1), contiene funciones, las cuales permiten capturar la secuencia de imágenes en tiempo real detectando los dispositivos de video conectados a la computadora, y sus parámetros correspondientes. El bloque de captura de video abre, inicializa, configura y controla un dispositivo de adquisición de video. La apertura, inicialización y configuración ocurre antes de la ejecución del modelo. El bloque tiene tres salidas que corresponden a los canales de color del video, rojo, verde y azul (R'G'B'). El bloque de video soporta solo dispositivos para Windows compatibles con DirectX. También es posible cambiar el módulo de captura por un módulo de lectura de un archivo de video (archivo tipo AVI), en el caso de que se requiera analizar un video previamente grabado. En este proyecto, la cámara utilizada es una cámara web, de la marca *Logitech QuickCam Express*. Esta cámara no cuenta con funciones especiales de captura o almacenamiento de datos, es simplemente un sensor de video (ver Figura 3.9).

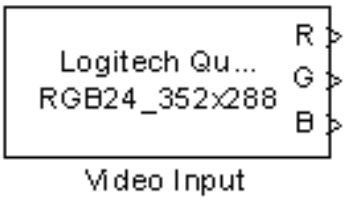


Figura 3.9 Bloque para captura de video de la plataforma MATLAB 2006a

Parámetros del bloque de captura de video (video Input): El bloque de captura de video cuenta con cuatro parámetros para seleccionar de acuerdo al tipo de video que se desea capturar. Los dos primeros parámetros corresponden al dispositivo de captura detectado (ver Figura 3.10).

Figura 3.10. Parámetros del bloque para entrada de video

Dispositivo: El sistema operativo de la computadora detecta los dispositivos de adquisición de imagen conectados a la misma. Por medio de este parámetro se selecciona uno de ellos (ver Figura 3.11).

Figura 3.11. Nombre de los dispositivos detectados.

Formatos de la entrada de video: Por medio de este parámetro se selecciona uno de los formatos de video soportados por el dispositivo de adquisición de imagen activado (ver Figura 3.12).

Figura 3.12. Formatos de video soportados por el dispositivo de video activado.

Velocidad de cuadros: Aquí se selecciona la velocidad expresada en cuadros por segundo (fps), a la cual se envían los cuadros al bloque de captura. Esta velocidad depende de la cámara con la que trabaje. La velocidad mínima de captura de la cámara aplicada, es de 5 cuadros por segundo y la máxima de 30 cuadros por segundo (ver Figura 3.13). Esta velocidad es afectada por el software de captura y el equipo de cómputo utilizado.

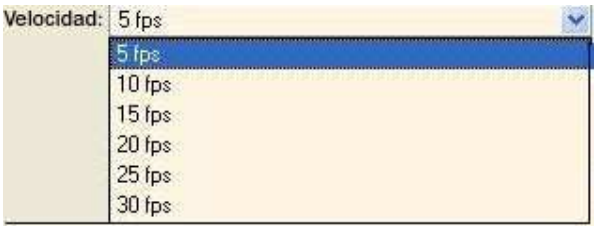


Figura 3.13. Velocidad de captura del dispositivo de video.

Tipo de dato de salida: El tipo de dato de la imagen es utilizado cuando el bloque envía cuadros de salida. Este tipo de datos indica cómo se almacenan internamente los cuadros de imágenes (ver Figura 3.14).

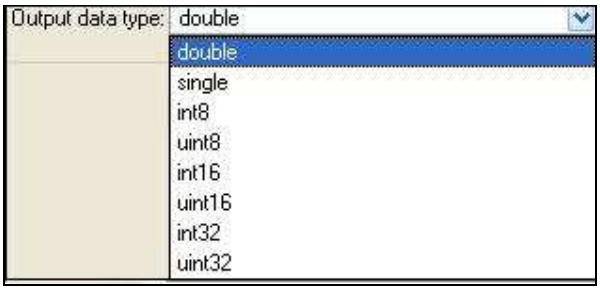


Figura 3.14. Tipo de datos con el que se almacenan internamente las imágenes.

3.2.3 Bloque de conversión de la información del color (R'G'B') a intensidad

En este bloque se convierte la información de color capturada por el dispositivo de adquisición de video a intensidad (ver Figura 3.15), es decir, convierte la imagen a niveles de gris, ya que el cálculo de flujo óptico se obtiene en base a la detección de cambios de niveles de gris entre dos imágenes.

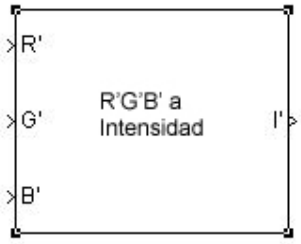


Figura 3.15. Bloque de conversión de parámetros R'G'B' a Intensidad.

Este bloque no sólo convierte de R'G'B' a intensidad, también permite la conversión entre los diversos parámetros del color existente (ver Figura 3.16). Todas las conversiones soportan punto flotante de precisión simple y doble. La conversión de R'G'B' a intensidad soporta además, entradas de datos enteros de 8 bits sin signo.

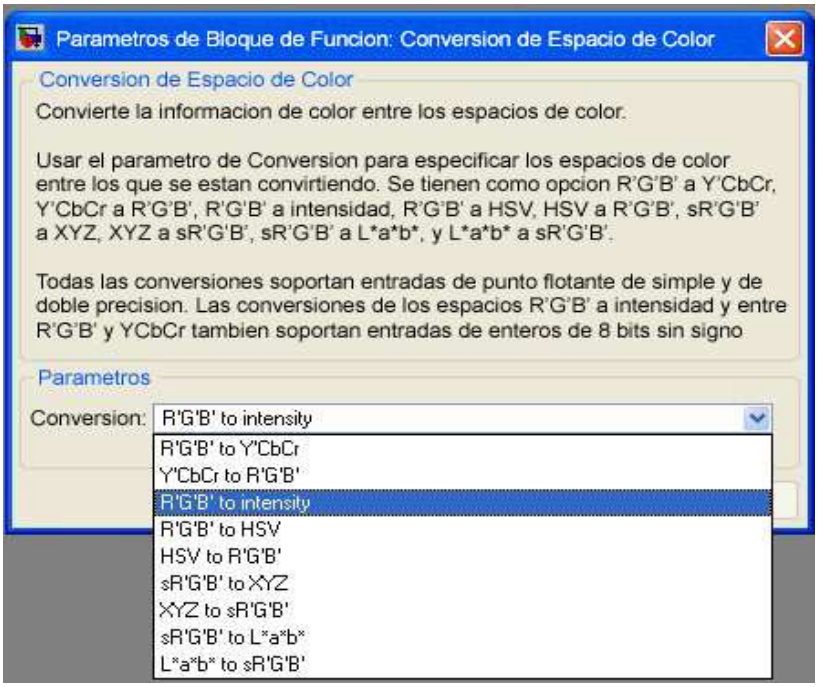


Figura 3.16. Parámetros de conversión entre los cuales se puede cambiar una imagen.

La conversión de los parámetros de color R'G'B' a intensidad se define por la siguiente ecuación:

$$I = \begin{bmatrix} 0.299 & 0.587 & 0.114 \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} \tag{3.8}$$

3.2.4 Bloque de cálculo de flujo óptico

Este módulo aplica la ecuación restringida de flujo óptico utilizando los métodos de Horn y Schunck o bien los algoritmos de Lucas y Kanade para la determinación de los parámetros de movimiento. La función de este bloque permite seleccionar tanto el método de Horn-Schunck como el algoritmo de Lucas-Kanade (ver Figura 3.17).

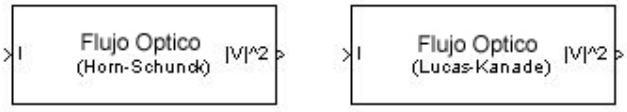


Figura 3.17. Bloques para el cálculo de flujo óptico según el algoritmo de Horn-Schunck o Lucas-Kanade.

Los bloques para cálculo de flujo óptico proporcionan determinados parámetros a manipular, los cuales dependen del algoritmo seleccionado (ver Figuras 3.18 y 3.19). A continuación se describen los parámetros que presenta cada método.

3.2.4.1 Parámetros correspondientes al bloque de cálculo de flujo óptico según el algoritmo de Horn-Schunck

Figura 3.18. Parámetros para el método de Horn-Schunck.

Selección de cuadros para el cálculo del flujo óptico *(Compute optical flow between)*: Con este parámetro se especifica si se calcula el flujo óptico entre dos imágenes *(Two images)* o en una secuencia de imágenes de *n* cuadros *(Current frame and Nth frame back)*. Si se selecciona este último como dato del parámetro, aparecerá como siguiente parámetro *N,* el cual corresponde a un valor escalar que representa el número de cuadros entre el cuadro de referencia y el cuadro actual.

Salida de velocidad ***(Velocity output).*** Por medio de este parámetro se específica el tipo de salida del bloque para cálculo de flujo óptico. Si se selecciona magnitud cuadrada *(Magnitud-squared),* la salida está dada por $u^2 + v^2$. Si se seleccionan componentes horizontales y verticales en forma compleja *(Horizontal and vertical components in complex form)*, la salida será $u + jv$.

Factor de suavizado ***(Smoothness factor)***. Como se ha mencionado anteriormente, el algoritmo de Horn-Shunck supone que los cambios de intensidad entre un cuadro y otro son suaves, por lo tanto, agrega un factor de suavidad, el cual varía de acuerdo a las condiciones de iluminación de la escena. Si el movimiento relativo entre dos imágenes o cuadros es grande, se requiere incorporar un valor escalar positivo grande. Si el movimiento relativo es pequeño, se requiere incorporar un valor escalar positivo pequeño. Este parámetro es utilizado solamente por el algoritmo de Horn-Schunck.

Solución iterativa de parada ***(Stop iterative solution).*** Este parámetro se utiliza para controlar el fin del proceso iterativo en el cálculo de flujo óptico. El proceso se detiene en tres formas:

1. Cuando la diferencia de la velocidad es menor que cierto valor de umbral *(When velocity difference falls below threshold)*. Por lo tanto, se tiene que especificar el umbral contra el cual se comparará el diferencial de velocidad.

2. Después de cierto número de iteraciones *(When maximun number of iterations is reached.),* entonces, se especifica el número máximo de iteraciones que el proceso deberá realizar.

3. Cuando ocurre cualquiera de las dos formas de finalización anteriormente mencionadas. Por consiguiente, se debe establecer tanto el umbral de comparación como el número máximo de iteraciones.

3.2.4.2 Parámetros correspondientes al bloque de cálculo de flujo óptico según el algoritmo de Lucas-Kanade

Cuando se selecciona el algoritmo de Lucas-Kanade para el cálculo del flujo óptico, los parámetros a especificar, son cuatro solamente (ver Figura 3.19). Tres de estos parámetros, se utilizan también en el algoritmo de Horn-Shunck descrito en la sección 3.2.4.1:

1. Selección de cuadros para el cálculo del flujo óptico *(Compute optical flow between)*.
2. El número de cuadros entre la primera y segunda imagen *(N)*.
3. La salida de velocidad *(Velocity output)*.

El cuarto parámetro, es utilizado únicamente por el algoritmo de Lucas-Kanade y es el siguiente:

Umbral para la reducción del nivel de ruidos *(Threshold for noise reduction):* es un valor escalar que determina el umbral del movimiento entre cada imagen o cuadro de la secuencia. Cuanto más alto es el número, menos afectan los movimientos pequeños al cálculo de flujo óptico (ver Figura 3.19).

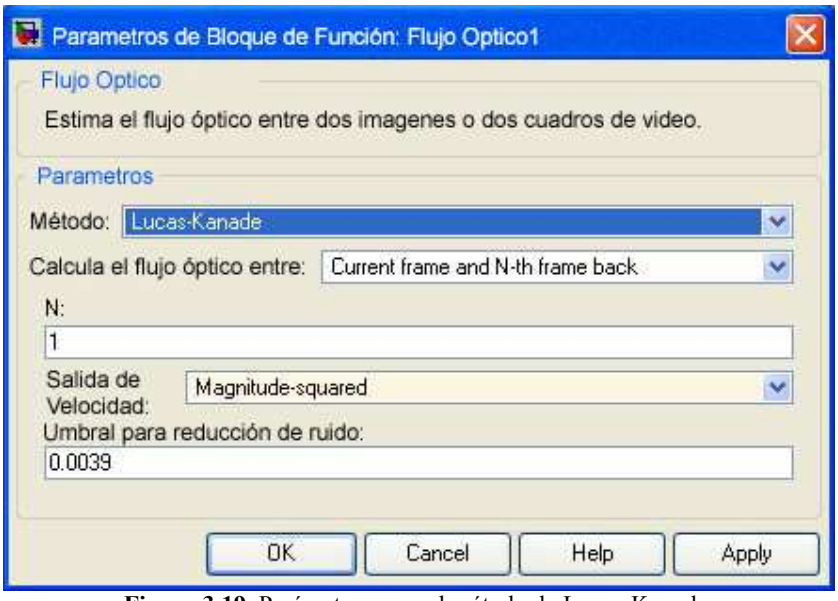


Figura 3.19. Parámetros para el método de Lucas-Kanade.

3.2.5 Bloques para umbralización y filtrado de la región de interés

En estos bloques se establece un umbral de imagen (ver Figura 3.20), para el cual son validas las magnitudes de flujo óptico calculadas por el bloque anterior; este valor depende de las características ambientales en la escena que se analiza, como la iluminación o los objetos de la escena. También incluye el proceso de filtrado de la región de interés (ROI). El proceso de filtrado se realiza mediante el bloque del análisis, el cual calcula ciertas características en una imagen binaria. El bloque devuelve algunos valores tales como el centro de figura, coordenadas espaciales o las cantidades de regiones de interés encontradas.

En la Figura 3.20, se observan los componentes internos para la umbralización y filtrado de la ROI. Este bloque tiene como entrada el valor de la magnitud cuadrada obtenida a partir del bloque de cálculo de flujo óptico. Esta magnitud es procesada por varios componentes internos, hasta obtener ROI umbralizadas y filtradas.

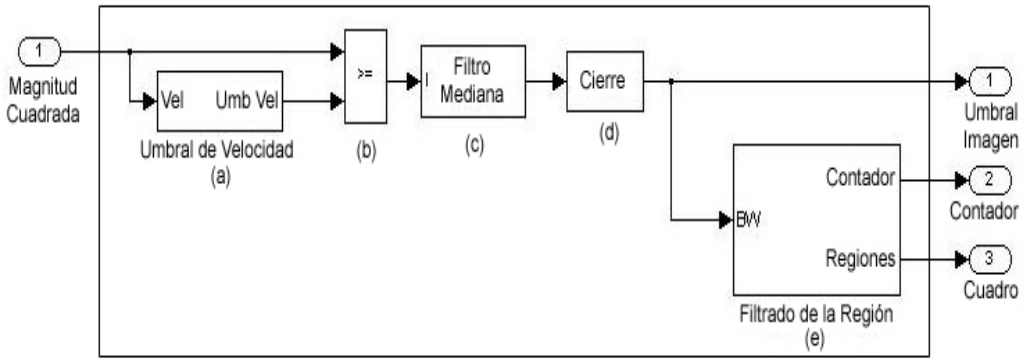


Figura 3.20. Diagrama interno del bloque de umbralización y filtrado de la ROI.

3.2.5.1 Umbral de Velocidad

El bloque de Umbral de Velocidad que se observa en el inciso (a) de la Figura 3.20, calcula la velocidad media por cuadro y la velocidad media por cuadro durante determinado tiempo transcurrido (ver Figura 3.21); también se calcula, la media sobre una ROI particular.

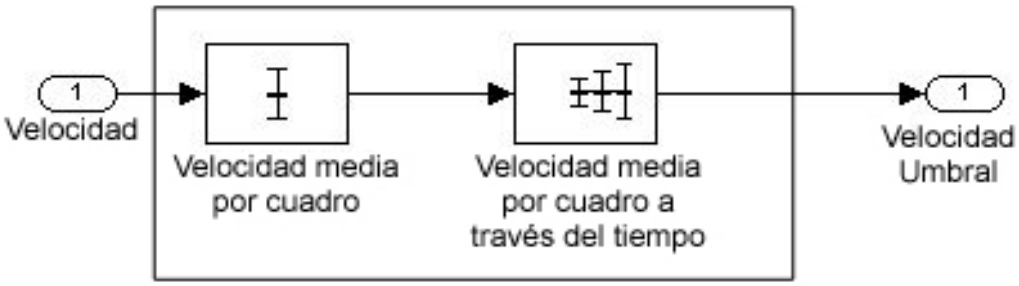


Figura 3.21. Diagrama interno del Umbral de Velocidad

3.2.5.2 Comparación

El Bloque de Comparación que aparece en el inciso (b) de la Figura 3.20, compara la magnitud cuadrada contra el umbral de velocidad, permitiendo el paso de aquellos valores que sean mayores o iguales al umbral obtenido y que correspondan a bordes (detección de cruce por cero).

3.2.5.3 Filtrado

El Bloque del Filtro de la mediana del inciso (c) de la Figura 3.20, substituye el valor central de la vecindad MxN, en este caso es de $3x3$, por su valor mediano. Si la vecindad tiene un elemento central, el bloque coloca el valor mediano en su lugar. Si la vecindad no tiene un centro exacto, el bloque coloca el valor mediano en una diagonal en la esquina superior izquierda (ver Figura 3.22). Para este bloque se eligió una vecindad con un elemento central, debido a que la vecindad es definida de 3x3.

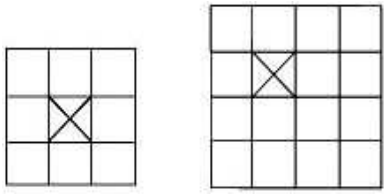


Figura 3.22. Vecindad con un elemento central y vecindad sin centro exacto

La salida del Bloque del Filtro, es una imagen de intensidad del mismo tamaño que la imagen de entrada.

3.2.5.4 Bloque de cierre

El Bloque de Cierre (d), realiza una operación de dilatación seguida por una operación de erosión (Soille, 2003) para definir la región de los movimientos a través de la imagen, usando una vecindad predefinida o un elemento morfológico estructural. En este bloque se crea un elemento de estructuración plano, linear, de longitud 3 a 45 grados (ver Figura 3.22). La longitud es aproximadamente la distancia entre los centros de los miembros del elemento de estructuración, en los extremos opuestos de la línea. Este bloque también entrega como salida la región umbralizada de los movimientos a través de la imagen, la cual será desplegada más tarde en una ventana de video (ver Figura 3.20) y conforma la entrada binaria (*Black and White*, BW) para el Bloque de Filtrado de la Región, donde se seleccionarán regiones de interés específicas de las que obtendremos los parámetros de movimiento (dirección y magnitud).

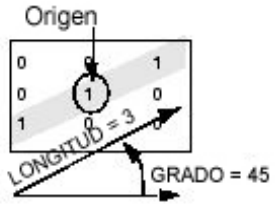


Figura 3.23. Elemento de estructuración plano, linear, de longitud 3 a 45 grados

3.2.5.5 Filtrado de la Región

El Bloque de Filtrado de la Región obtiene como salida un recuadro de la ROI detectada y un contador de regiones en cada objeto binario procesado (*Binary Large OBject,* BLOB). El proceso interno del bloque se estructura como se observa en la Figura 3.24.

El primer Bloque de Análisis (ver Figura 3.24 (a)) recibe la señal binaria umbralizada en la etapa anterior. Analiza esta señal para detectar BLOBs que cumplan con determinados criterios de selección de acuerdo al tipo de ROI que se desea localizar. Los criterios o parámetros a establecer de dichos objetos son: cantidad máxima de objetos y área mínima y máxima de los objetos, dada en píxeles. Una vez localizados los objetos, el bloque devuelve valores que representan la localización de la ROI. Los valores obtenidos son centroídes, áreas específicas o bordes. En este modelo, los valores de interés son los bordes de dichos objetos, puesto que permiten reducir los cálculos y

ahorrar recursos computacionales, así como descartar el fondo de la imagen de los objetos que están en movimiento.

El Bloque Selector de Variables (ver Figura 3.24 (b)), extrae un subconjunto de filas o de columnas de *MxN* en la matriz de entrada que proviene del bloque de Análisis. Selecciona y ordena estas filas o columnas de acuerdo a un vector específico de índices que se recibe por el puerto *Idx* . Los parámetros a establecer en este bloque son el número de entradas y puertos de salida de acuerdo a la ROI a detectar; determinar si el conjunto de datos a extraer de cada matriz son las columnas o las filas; por último, determinar el modo de selección de los valores del vector, es decir, si el bloque utiliza los mismos índices para cada entrada (valores fijos), o diferentes (variables). En este caso, se determina una sola entrada (Entrada1) que recibe la matriz de datos del BLOB obtenido por el bloque de Análisis. Por lo tanto, el puerto de entrada *Idx ,* recibe la longitud del vector de índices que se quiere detectar (ver Figura 3.24 (c) y (d)) en la matriz de entrada, es decir, índices variables que cumplan con determinado rango, de tal modo que la salida final del bloque es las ROI detectadas con los parámetros establecidos. Estos parámetros se cambian de acuerdo a la ROI de la que se desea obtener el flujo óptico.

El Bloque de la Submatriz (ver Figura 3.24 (c)), selecciona un subconjunto de elementos de la matriz de entrada para formar la matriz con un determinado rango de columnas y renglones. El Bloque de Comparación (ver Figura 3.24 (d)), compara la señal de entrada con una constante, para filtrar las ROI de un tamaño determinado, dependiendo del área que se desea detectar. Con el bloque de Tipo de dato (ver Figura 3.24 (e)) se convierte la entrada al tipo de dato que se utiliza como salida: punto flotante de precisión simple o doble, enteros con signo o sin signo de 8 16 y 32 bits y boléanos. En este caso se requiere convertir los valores de las áreas seleccionadas, a datos punto flotante de precisión simple. Los datos punto flotante, almacenan aproximaciones de números reales para ahorrar consumo de memoria y conservar los datos de la matriz lo más exacto posible. El Bloque de la Suma (ver Figura 3.24 (f)) realiza la adición o la substracción en sus entradas. Con este bloque se están contabilizando las ROI encontradas en cada BLOB analizado. Por último, nuevamente se utiliza un Bloque de Tipo de dato (ver Figura 3.24 (g)) para convertir la ROI a datos de 32 bits e imprimirla en el recuadro de resultados.

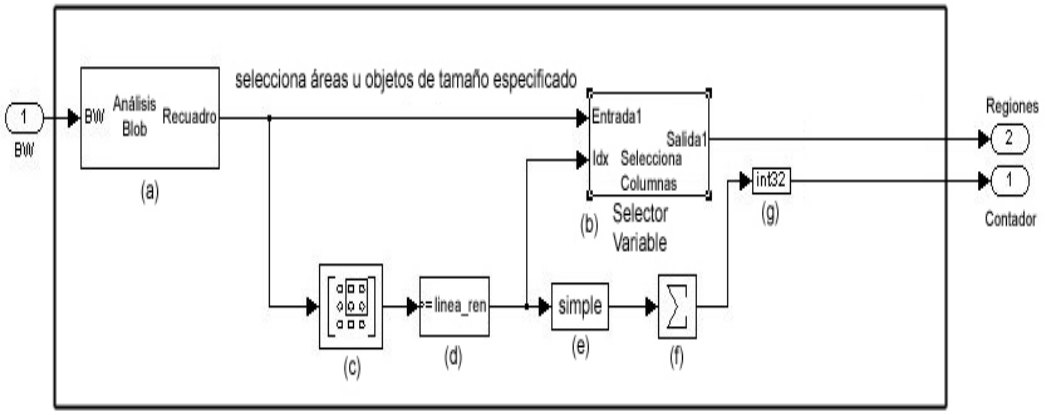


Figura 3.24. Diagrama interno del bloque de Filtrado de región.

3.2.6 Bloque de despliegue de resultados

En esta última etapa se despliegan los resultados finales obtenidos del proceso de cálculo de flujo óptico en tiempo real. Los resultados se muestran en tres diferentes ventanas (ver Figura 3.25). La primera ventana de Umbral, recibe una imagen de intensidad del mismo tamaño que la imagen original la cual muestra el umbral de cada cuadro de video capturado; la segunda ventana de Resultados, muestra la cantidad de regiones encontradas en la secuencia cada determinado tiempo y enmarca las regiones en un rectángulo al momento de ser detectadas sobre el video real; la tercera ventana despliega el video Original, esto con la finalidad de comparar robustamente los resultados obtenidos del proceso.

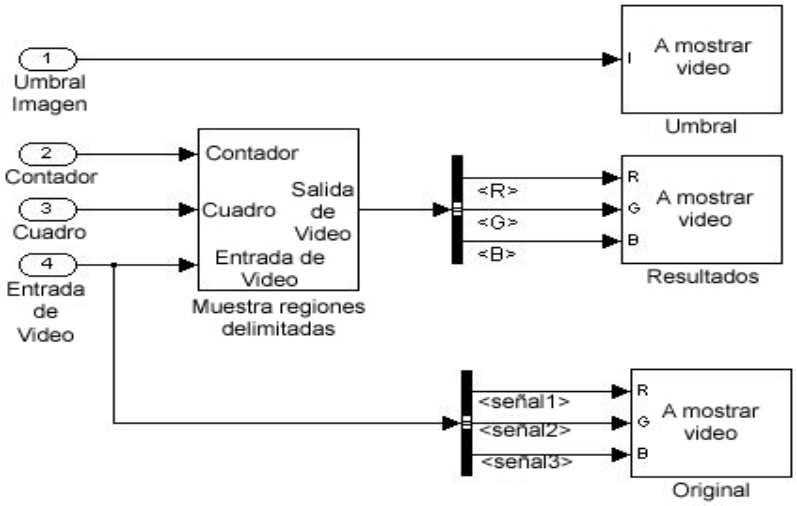


Figura 3.25. Diagrama interno del bloque de Despliegue de Resultados.

El Bloque de Dibujar formas (ver Figura 3.26), dibuja rectángulos, líneas, polígonos, o círculos sobreescribiendo los valores de los píxeles en las imágenes. Consecuentemente, las formas se insertan en la imagen de salida. Este bloque utiliza el algoritmo de Bresenham's (Gupta y Sproull, 1981) para trazar líneas, polígonos, círculos y rectángulos. En este caso, se trazan rectángulos sobre las áreas en movimiento, debido al interés de observar gráficamente la localización espacial de dichas áreas. El tamaño de estas áreas se recibe a partir de la salida de Cuadro de la etapa de filtrado de la región. El inconveniente de este bloque es que sólo permite seleccionar una figura ya definida.

Los Bloques siguientes, sobreescriben datos en cada submatriz de la señal de video R'G'B' para definir el espacio de la imagen a partir del cual serán detectadas las ROI y el espacio de la imagen donde se imprimirá el contador de regiones o áreas en movimiento. Por último, el bloque de Insertar texto (ver Figura 3.26) imprime el texto o números en el espacio definido por los bloques anteriores. La salida en esta etapa es finalmente la delimitación del espacio de detección de las regiones o áreas en movimiento y la impresión de rectángulos sobre las regiones en movimiento detectadas, para ser observadas en la ventana de despliegue de resultados.

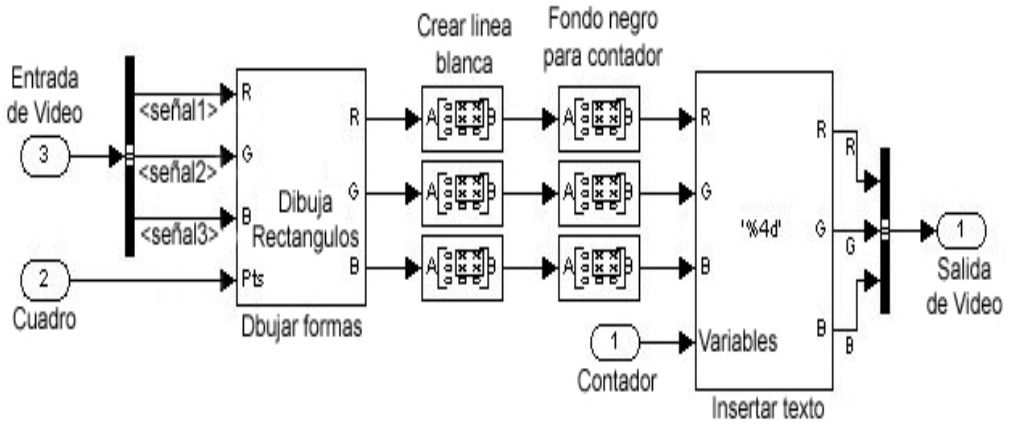


Figura 3.26. Diagrama interno del bloque de Muestra de regiones delimitadas.

3.2.7 Conclusiones de la etapa de modelado

En la segunda etapa, se diseña un modelo basado en las fases establecidas por Horn-Schunck, para la determinación del flujo óptico en una secuencia de imágenes. Estas fases se describen en el capítulo II. El modelo permite el análisis de los dos algoritmos principales para el cálculo de flujo óptico, Horn-Schunck y Lucas Kanade. Con el modelado a bloques de procesamiento de imágenes y video de *Simulink*, se diseñó el sistema de cálculo de flujo óptico en tiempo real con las siguientes características: una velocidad mínima de 5 cuadros por segundo y una máxima de 15 cuadros por segundo. Aunque la velocidad máxima de captura de la cámara es de 30 cuadros por segundo, esta se reduce a la mitad, debido a que MATLAB es un software que consume gran cantidad de recursos computacionales. MATLAB 2006a es un software que requiere ciertas características del hardware para su funcionamiento mínimo: 512 Megabytes de RAM, más de 1 Giga Hertz de velocidad en el procesador, además de un espacio libre de disco duro de 510 Mega Bytes. A pesar del inconveniente de la velocidad, con este modelo se analizó el comportamiento del método de Horn-Schunck y Lucas-Kanade, en tiempo real. Se determina que el procesamiento con el algoritmo de Horn-Schunck, consume más recursos computacionales que el método de Lucas-Kanade, provocando un tiempo de ejecución mayor. Se detecta mayor cantidad de puntos de flujo óptico falsos en el algoritmo de Horn-Schunck. Con el método de Lucas-Kanade esta detección de puntos de flujo óptico falsos es considerablemente menor, debido al umbral para la reducción del nivel de ruidos τ .

La herramienta para modelado de MATLAB 2006a tiene el inconveniente carecer de flexibilidad para realizar modificaciones o aportaciones al código y la mayoría de los parámetros están sujetos a determinados rangos.

CAPÍTULO IV

SISTEMA PARA CÁLCULO DE FLUJO ÓPTICO

Durante la etapa de modelado se analizaron cualitativamente parámetros determinantes para seleccionar uno de los dos algoritmos principales (Horn-Schunck y Lucas-Kanade), para implementarlo como base en el desarrollo de sistema. Este algoritmo es el de Lucas-Kanade, por sus resultados satisfactorios en cuanto a sensibilidad al ruido, localización óptima de puntos característicos para el cálculo de los vectores de flujo óptico y optimización de recursos computacionales. La implementación está basada en el modelo definido en la segunda etapa de implementación expuesta en el capítulo III. Dicho modelo se mejora al combinar el algoritmo de Lucas-Kanade con pirámides Gaussianas y seleccionar un algoritmo previo para localizar puntos característicos y regiones de interés que facilitan el cálculo de los vectores de flujo óptico dentro de una escena dinámica tal como se explica en (Shi y Tomasi, 1994).

4.1 Fases propuestas para el cálculo de flujo óptico por gradientes

Las fases para el cálculo de flujo óptico por gradientes consisten en una serie de pasos básicos, aplicados a una secuencia de imágenes. Horn y Schunck (1981) establecen cuatro fases para la estimación del flujo óptico. Estas fases se exponen en el capítulo II de esta tesis. Para el desarrollo del Sistema para Cálculo de Flujo Óptico, se establece un cambio en dichas fases. El cambio consiste en la selección de puntos característicos antes de la detección de flujo óptico básico, lo cual permite eliminar la fase de aplicación de restricciones de suavidad en cambios en el flujo óptico obtenido. Este cambio, reduce el tiempo de procesamiento de datos, debido a que el flujo óptico se calcula sólo en las áreas determinadas por el algoritmo de selección de puntos característicos, lo cual evita la aplicación de restricciones posteriores, para eliminar valores no válidos de vectores de flujo óptico calculados. Las fases establecidas por Horn y Schunck y las fases propuestas en esta tesis, se observan en la Tabla 4.1, donde la diferencia entre ambas se indica con una flecha.

Tabla 4.1. Fases para el cálculo de flujo óptico por gradientes

Fases establecidas por Horn y Schunck	**Fases aplicadas al Sistema para Cálculo de Flujo Óptico**
1. Suavizado de la secuencia. 2. Cálculo de gradientes. 3. Determinación del flujo óptico básico. ⇨ 4. Aplicación de restricciones de suavidad en cambios en el flujo óptico.	1. Suavizado de la secuencia. 2. Cálculo de gradientes. ⇨ 3. Selección de puntos característicos 4. Determinación del flujo óptico básico.

4.2. Selección de puntos característicos.

El flujo óptico se define como los cambios de intensidad en una secuencia de imágenes, provocados por el movimiento de un punto. Por consiguiente, el seguimiento de puntos de una imagen a otra, es la base de los algoritmos para cálculo de flujo óptico. Para estos algoritmos resulta inconveniente el cálculo de flujo óptico en cada punto de la imagen porque no todos los puntos corresponden a parámetros de magnitud y dirección del movimiento, debido a las restricciones de la ecuación para el cálculo de flujo óptico. Se deben seleccionar los puntos característicos de acuerdo a la aplicación a la que se destine el sistema o a la precisión, robustez y funcionalidad del seguimiento. Las características que deben reunir estos puntos de interés se resumen en cuatro propiedades (Hartley, 85; Harris, 88):

1. **Distinción,** significa que un punto debe ser diferente de sus vecinos inmediatos. Esto excluye la selección de puntos pertenecientes a áreas uniformes de la imagen o bordes rectilíneos, ya que las distintas partes de un borde rectilíneo no pueden diferenciarse entre sí. Es lo que se conoce como "problema de la apertura" *(aperture problem).*

2. **Unicidad,** significa que un punto debe ser distinguible globalmente, es decir, idealmente no debe parecerse a ningún otro punto de la imagen. Para asegurar

que los puntos cumplan esta propiedad debe evitarse la selección de elementos que, aunque resulten localmente distinguibles, aparezcan repetidamente en la imagen. Este tipo de puntos repetidos afecta negativamente a los procesos de búsqueda de correspondencias, al provocar situaciones de confusión.

3. **Invarianza,** se refiere a que la apariencia de un punto, no debe variar a consecuencia de las distorsiones geométricas o radiométricas ocurridas, debido a las características del objeto y su movimiento o en relación con la iluminación de la escena o el proceso de formación de la imagen.

4. **Estabilidad,** se refiere a que la apariencia del punto debe ser invariante respecto al punto de vista. Puntos interesantes de la imagen deben corresponder a puntos de interés del objeto. Deben excluirse puntos que resultan del cruce de bordes de distintos objetos o de un objeto y el fondo.

Existen algunos procedimientos para la selección automática de puntos característicos los cuales buscan puntos con buena distinción, basándose en alguna medida de varianza local de la imagen o en alguna característica como la calidad de esquina o borde. En el desarrollo del Sistema para Cálculo de flujo óptico, se utiliza uno de los procedimientos para la selección automática de puntos característicos, basado en la mejora de la estimación del desplazamiento conocido como movimiento afinado desarrollado por Jianbo Shi y Carlo Tomasi (1994). Este método propone un criterio de selección óptimo, capaz de detectar oclusiones, desoclusiones y descartar características que no corresponden al movimiento, además está basado en los trabajos de detección de movimiento realizados por Lucas y Kanade (1981). El método considera la minimización de un criterio de similitud basado en sumas de diferencias al cuadrado entre los píxeles de una pequeña ventana alrededor del punto candidato en los dos cuadros de imagen consecutivos (Lucas y Kanade, 81). El método aplica dos casos: en el primero asume que la ventana se desplaza con una simple traslación (se considera que la matriz de píxeles no sufre deformación alguna) tal y como lo afirman Lucas y Kanade en su investigación; en el segundo se establece que la ventana puede sufrir una transformación afín de un cuadro al siguiente que, además de considerar su traslación, intenta modelar la posible deformación sufrida. Para el caso más simple, el de traslación, la determinación del vector de desplazamiento d sufrido por un punto de una imagen f_1 a la siguiente imagen f_2 se realiza minimizando la expresión:

$$Gd = e, \tag{4.1}$$

donde la matriz de coeficientes G es la matriz $2x2$ simétrica:

$$G = \int_w g g^T \omega dA, \tag{4.2}$$

siendo g el vector de gradiente de la imagen f_1 , W la ventana considerada y ω es una función de ponderación. En el lado izquierdo de la expresión (4.1), e es el vector bidimensional:

$$e = \int_w (f_1 - f_2) g \omega dA. \tag{4.3}$$

Para que el sistema (4.1) se resuelva de forma fiable, la matriz $2x2$ de coeficientes G del sistema debe estar por encima del nivel de ruido y debe ser además bien condicionada. El primer requerimiento implica que ambos valores propios de G (λ_1, λ_2) deben ser grandes, mientras que el requerimiento sobre su condicionamiento significa que estos valores propios no deben diferir entre sí en varios órdenes de magnitud. Como en la práctica los valores de intensidad en la ventana están limitados por el máximo valor de píxel permitido, el mayor de los valores no debe ser arbitrariamente grande y este segundo requerimiento no es decisivo. Por ello, considerando el primer requerimiento respecto al ruido, se establece el siguiente criterio como condición de selección de una determinada ventana como punto de interés:

$$mín\,(\lambda_1, \lambda_2) > \lambda, \tag{4.4}$$

donde λ es un umbral predefinido. Para determinar el umbral λ se miden primero los valores propios para zonas de la imagen con brillo uniforme. Ello da un límite inferior para λ. A continuación se selecciona un conjunto de diferentes tipos de puntos característicos, como esquinas o regiones de alta textura, para obtener un límite superior de λ. En la práctica se escoge como valor para λ el valor medio de ambos límites.

De acuerdo a los criterios de este método, una ventana que ofrece dos pequeños valores propios para la matriz G corresponde a una región de imagen con un perfil de intensidad prácticamente constante y este no es seleccionado. Tampoco se seleccionan ventanas que tengan un alto valor para uno de los valores propios, pero un valor

pequeño para el otro, lo que corresponde a ventanas con un patrón de intensidad orientado en una única dirección. En cambio, dos valores altos propios suelen corresponder a esquinas o regiones de alta textura con distribución no orientada y dan lugar a que la ventana se seleccione. Podemos concluir que este es un método bien condicionado e insensible a ruido.

4.3 Requerimientos básicos del sistema

Para la realización de este proyecto, se utiliza como hardware, una cámara Web de baja resolución y bajo costo, una computadora Intel Celeron de 1.70 Giga Hertz con una capacidad de memoria de 256 mega bytes. Para el desarrollo del programa se emplea Lenguaje "C", ya que es funcional en diversas plataformas, además se incluye la biblioteca de funciones para procesamiento de imágenes conocida como OpenCV de la compañía Intel, por su funcionalidad para trabajar en tiempo real y por ser software de distribución libre (OpenCV, 2000-2006).

4.4 Funcionalidad del sistema

El programa se diseña para un sistema monocular y cuenta con las siguientes funciones:

- Captura de video en tiempo real, la cual se realiza tanto en interiores como en exteriores debido a la modificación de los parámetros de iluminación y de la selección de puntos característicos, mediante el algoritmo de seguimiento de partículas de Shi y Tomasi (1994).
- Cálculo de la magnitud y dirección del movimiento.
- Detección de la magnitud mayor.
- Impresión gráfica del campo de flujo óptico sobre el video capturado.
- Cálculo de la velocidad de procesamiento en cuadros por segundo.

Estas funciones son modulares y se añaden en la implementación de aplicaciones específicas. Esta modularidad se comprueba al incluir las funciones desarrolladas de detección de movimiento, conjuntamente con las funciones de reconocimiento de patrones en una aplicación para el seguimiento de rostros y en un instrumento para análisis cuantitativo del movimiento de microestructuras.

4.5 Estructura del sistema

El programa principal del Sistema para Cálculo de Flujo Óptico se estructura en base al modelo diseñado con MATLAB, el cual es definido en el capítulo III de esta tesis. El modelo se constituye por cinco bloques principales tomando en cuenta que son éstas, las etapas básicas que conforman un sistema de visión artificial. Las funciones de los bloques corresponden a los procesos de captura de imágenes, conversión de R'G'B' a intensidad, cálculo de flujo óptico, umbralización y despliegue de resultados. Estos procesos se incluyen en el programa principal del sistema.

El sistema para cálculo de flujo óptico se desarrolla con Lenguaje "C". Está basado en las fases para cálculo de flujo óptico por gradientes, propuestas en esta tesis. Los algoritmos correspondientes a estas fases se indican en la Figura 4.1. Se incluye como algoritmo de seguimiento de puntos característicos el establecido por Shi y Tomasi en 1994, ya que es un algoritmo general capaz de resolver problemas de oclusión y eliminación de puntos no correspondientes a la escena, logrando con ello el ahorro de recursos computacionales.

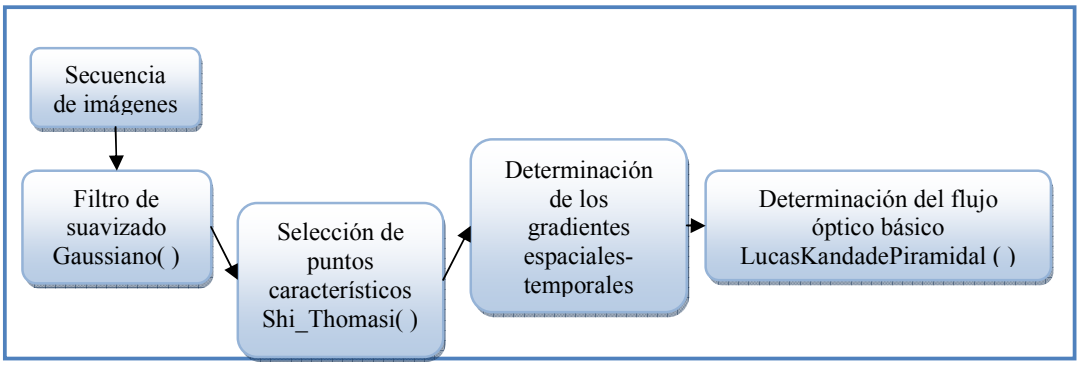


Figura 4.1. Fases para cálculo de flujo óptico por gradientes aplicadas al Sistema

4.6 Funciones principales del Sistema

El Sistema para cálculo de Flujo Óptico contiene ocho funciones principales:

a) Inicialización de los dispositivos de video: Esta función detecta y activa los dispositivos de captura del video.

b) Creación de ventana de resultados: Se crean las ventanas de visualización de resultados.

c) Captura de cuadros: Esta función toma dos cuadros consecutivos de la secuencia de video su procesamiento.

d) Conversión de cuadros a niveles de gris: En esta función, las imágenes a color se convierten a imágenes en niveles de gris, porque los cambios de intensidad en los niveles de gris dan como resultado la detección de flujo óptico.

e) Aplicación del filtro de suavidad. Se filtran las imágenes para reducir el nivel de ruido y los cambios bruscos de niveles de gris.

f) Algoritmo de seguimiento de características de Shi-Thomasi: Esta función localiza los puntos validos de las imágenes para el cálculo del flujo óptico.

g) Cálculo de flujo óptcio por el algoritmo de Lucas-Kanade: Esta función determina los parámetros de flujo óptico de los puntos seleccionados.

h) Generación e impresión del campo de flujo óptico: Contiene un algoritmo para calcular y desplegar en las ventanas de visualización de resultados, la magnitud y dirección de los vectores de flujo óptico.

Las funciones desarrolladas se observan en el diagrama de flujo de la figura 4.2.

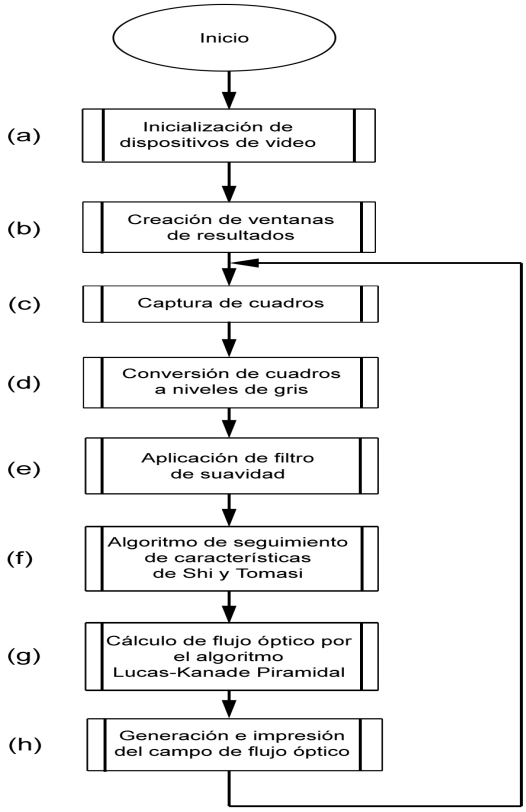


Figura 4.2. Diagrama de flujo del programa principal del sistema de cálculo de flujo óptico

4.6.1 Inicialización de dispositivos de video.

El algoritmo para la inicialización de video detecta y activa el sensor de video conectado a la computadora. Si no existe conexión alguna, se envía un archivo de video (archivo.avi) como parámetro para la detección de movimiento en un archivo pregrabado (ver Figura 4.3).

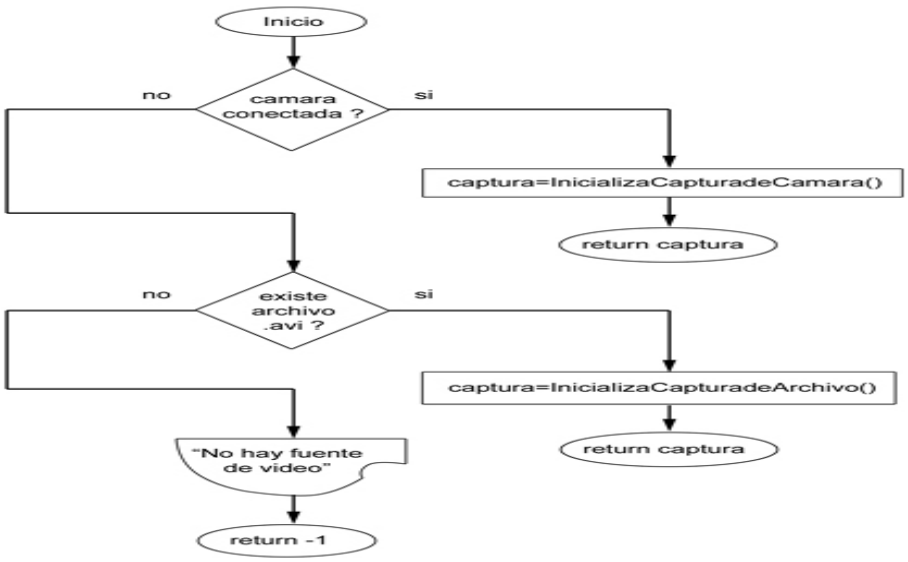


Figura 4.3. Diagrama de flujo de la función para inicializar y activar captura de video

4.6.2 Creación de ventanas de resultados

En este módulo se crean cuatro ventanas de visualización de resultados. La primera ventana muestra los cuadros del video original, esto con la finalidad de comparar con las ventanas que muestran proceso y resultados. La segunda y tercera ventana muestran los cuadros de intensidad (niveles de gris), anterior y actual, con los que se realizan los cálculos de flujo óptico. La cuarta ventana, muestra el campo de flujo óptico de modo gráfico sobre el video original, lo que permite observar, los vectores de movimiento en tiempo real presentes en la escena (ver Figura 4.4).

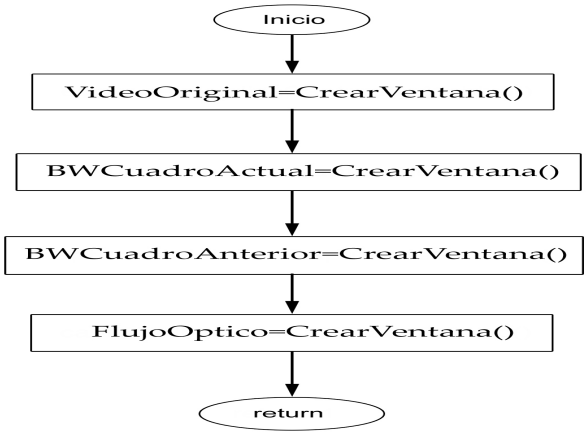


Figura 4.4. Diagrama de Flujo para la creación de ventanas de resultados

4.6.3 Captura de cuadros

Se crea una función para capturar dos cuadros de videos, los cuales se procesan posteriormente para calcular el flujo óptico. El primer cuadro se conoce como actual, que es el cuadro más reciente capturado. El segundo cuadro se identifica como anterior, el cual inicialmente se crea vacío. En cada vuelta de ciclo, el cuadro anterior toma los valores de cuadro actual, para que el cuadro actual tome los valores de una nueva imagen de tal modo que siempre existan dos cuadros para ser analizados. En el caso de no recibir un nuevo cuadro capturado, el proceso termina (ver Figura 4.5).

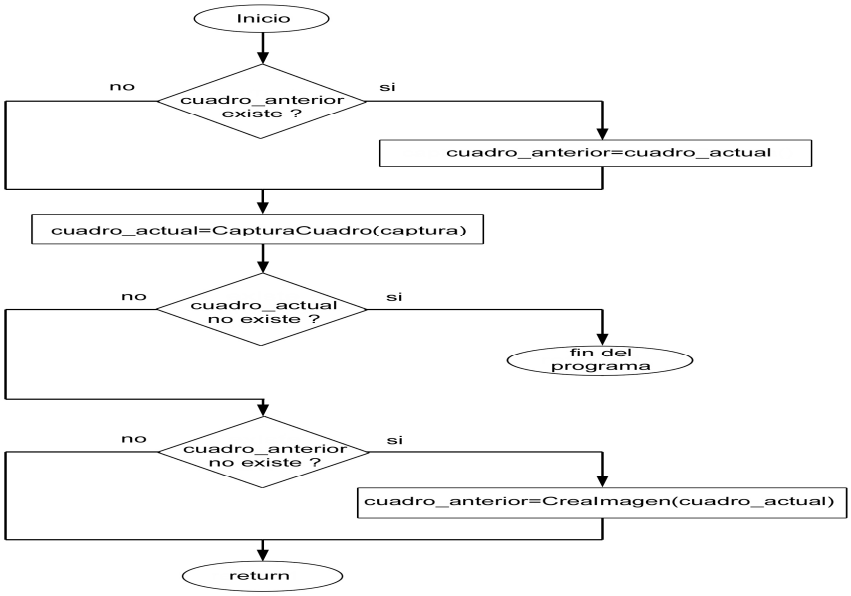


Figura 4.5. Diagrama de flujo para capturar dos cuadros de video

4.6.4 Conversión de cuadros a niveles de gris

Los cuadros capturados, provienen directamente del sensor de video o de un archivo de video en color. El procesamiento para el cálculo de flujo óptico consiste en el cambio de niveles de gris, por lo tanto, se requiere la conversión de estos cuadros de color (R'G'B') a niveles de gris (ver la Figura 4.6).

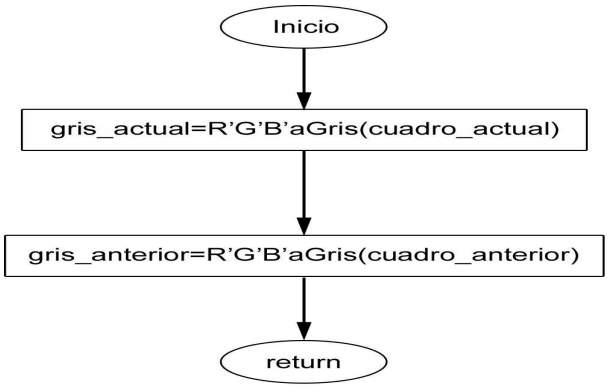


Figura 4.6. Diagrama de flujo para la conversión de cuadros R'G'B' a niveles de gris

4.6.5 Aplicación de Filtro de Suavidad

La información contenida en los cuadros de niveles de gris, aún requieren de un procesamiento más, antes de realizar los cálculos de flujo óptico. La matriz de datos de cada cuadro contiene elementos de ruido que provocan cálculos falsos en etapas posteriores. Para eliminar el ruido y cambios bruscos en los niveles de grises, se aplica un filtro Gaussiano a cada cuadro (ver Figura 4.7).

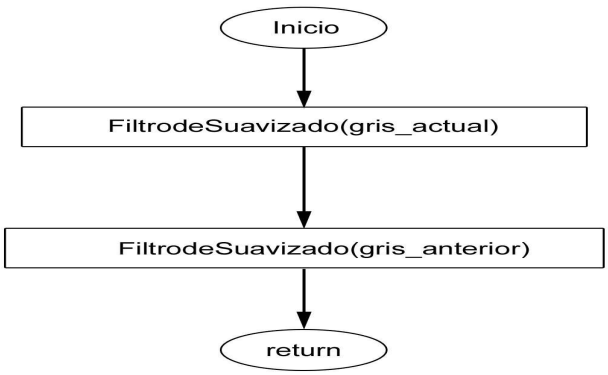


Figura 4.7. Aplicación de Filtro de suavizado a los cuadro de niveles de gris.

4.6.6 Algoritmo para el seguimiento de características

Calcular el flujo óptico sobre toda la imagen, resulta un proceso lento y existen restricciones para la detección de movimiento. Para aligerar el proceso se emplea un algoritmo que aplica un criterio para la selección óptima de características y su seguimiento de un cuadro a otro.

El algoritmo para el seguimiento de características, es el algoritmo establecido por Shi y Thomasi en 1994. Este algoritmo detecta oclusiones, desoclusiones y las características que no corresponden a los puntos en la escena. Además con este criterio, se seleccionan solo aquellas áreas, donde es posible la detección de movimiento, como bordes o esquinas, los cuales permiten distinguir el fondo de la escena, de los objetos existentes, reduciendo el proceso de cálculo de flujo óptico.

El algoritmo encuentra esquinas con valores propios en la imagen (puntos_gris_anterior). La función primero calcula el valor propio mínimo para cada píxel de la imagen fuente y los almacena en un vector (eig_image). Entonces realiza la eliminación de los no-máximos (solamente sigue habiendo los máximos locales en la vecindad 3x3).

El paso siguiente es rechazar las esquinas con el valor propio menor que los valores mínimos almacenados. Los valores no rechazados se multiplican por el factor de calidad que se quiere detectar (calidad*eig_image(x, y)).

Finalmente, la función se asegura de que todas las esquinas encontradas estén lo suficientemente separadas una de otra para ser consideradas válidas (las esquinas más fuertes se consideran primero) y comprobar que la distancia entre la nueva característica y las características anteriores es más grande que la distancia mínima establecida (distancia_mínima). De este modo, la función elimina las características que no cumplan con los rangos establecidos. Este proceso se observa en la Figura 4.8.

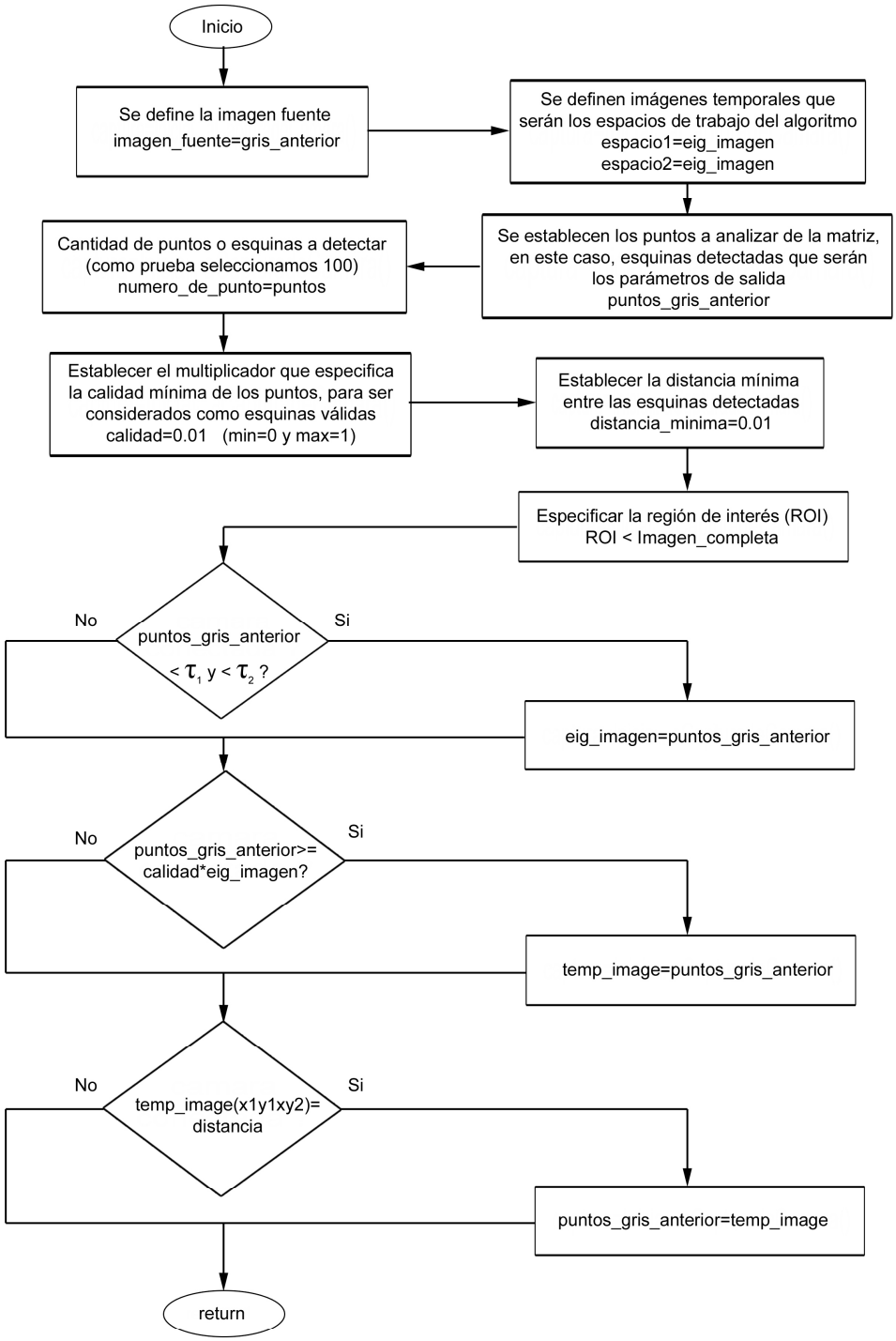


Figura 4.8. Diagrama de flujo del algoritmo para el seguimiento de características de Shi-Tomasi

4.6.7 Cálculo de flujo óptico por el algoritmo de Lucas-Kanade Piramidal

El algoritmo de Lukas-Kanade Piramidal, calcula las coordenadas de los puntos característicos en el cuadro de video actual dadas sus coordenadas en el cuadro anterior. La función encuentra las coordenadas con exactitud del píxel secundario. Los vectores para almacenar las pirámides del primer cuadro (pirámide_anterior) y segundo cuadro (pirámide_actual), se conforman con las siguientes reglas:

- Si el apuntador de la imagen es 0, la función asigna el vector internamente, calcula la pirámide y libera el vector después de procesar.
- Si el apuntador no es cero, la función calcula la pirámide y almacena los valores en el vector a menos de que se activen las banderas de precálculo de las pirámides (CV_LKFLOW_PYR_A[B]_READY) antes de la llamada a la función.
- La imagen debe ser lo suficientemente grande para almacenar los datos de la pirámide Gaussiana.
- Después de la llamada de función se calculan ambas pirámides y la bandera de precálculo para la imagen correspondiente se puede activar en la llamada siguiente (ver Figura 4.9).

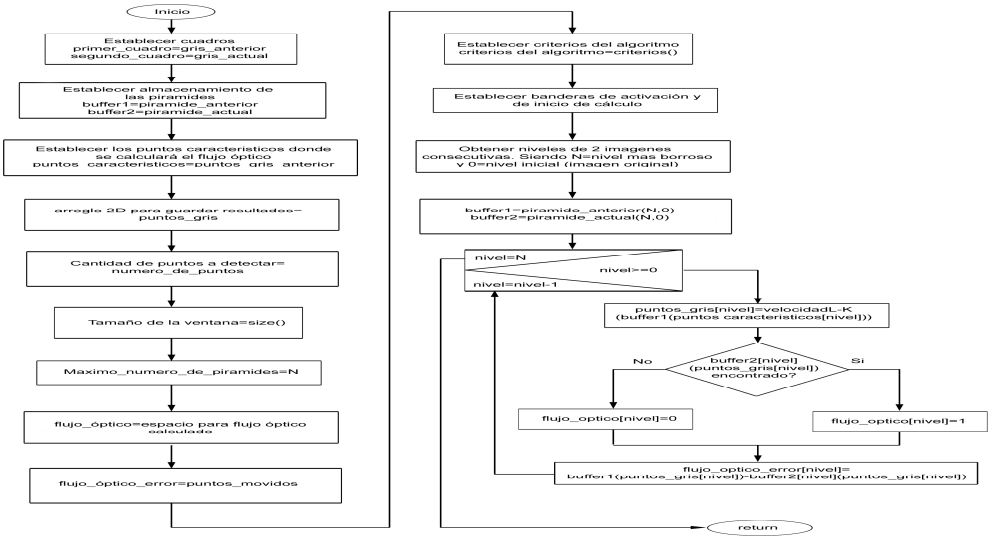


Figura 4.9. Diagrama de flujo del algoritmo de Lucas-Kanade Piramidal

4.6.8 Generación e impresión del campo de flujo óptico

Para la generación e impresión del campo de flujo óptico sobre la secuencia de video en tiempo real, se diseña un algoritmo (ver Figura 4.11) que utiliza como parámetros de entrada, los valores resultantes del proceso de cálculo de flujo óptico del algoritmo de Lucas-Kanade piramidal. Como puntos espaciales, se toman los puntos de la primera posición del vector de resultados de flujo óptico y los puntos del primer cuadro. Por medio de un ciclo que inicia desde la posición cero de los vectores, hasta el número de puntos procesados, se calcula el ángulo y magnitud de los puntos de flujo óptico almacenados en los vectores de resultados y se genera con ellos el campo de flujo óptico.

$$angulo = atan2((double) p.y - q.y, (double) p.x - q.x) \quad (4.5)$$

$$magnitud = sqrt((p.y - q.y)*(p.y - q.y) + (p.x - q.x)*(p.x - q.x)) \quad (4.6)$$

El campo de flujo óptico se visualiza sobre la secuencia de video en tiempo real (ver Figura 4.10).

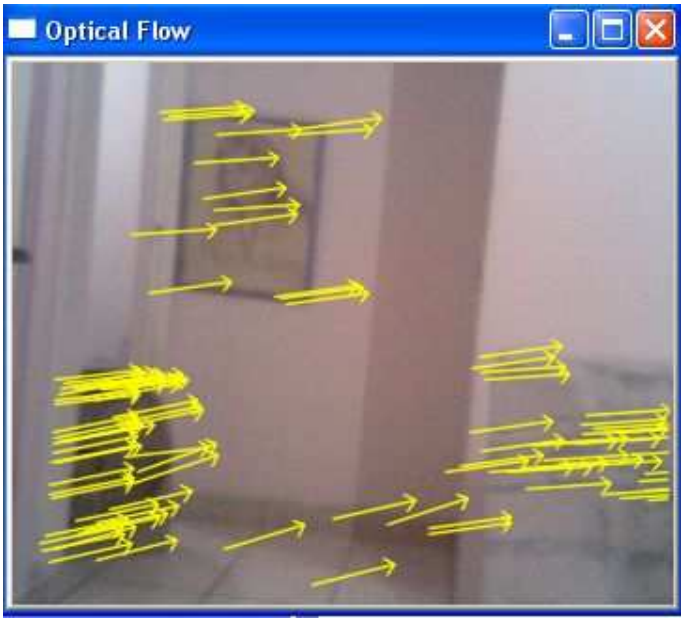


Figura 4.10. Campo de flujo óptico generado sobre la secuencia de video en tiempo real.

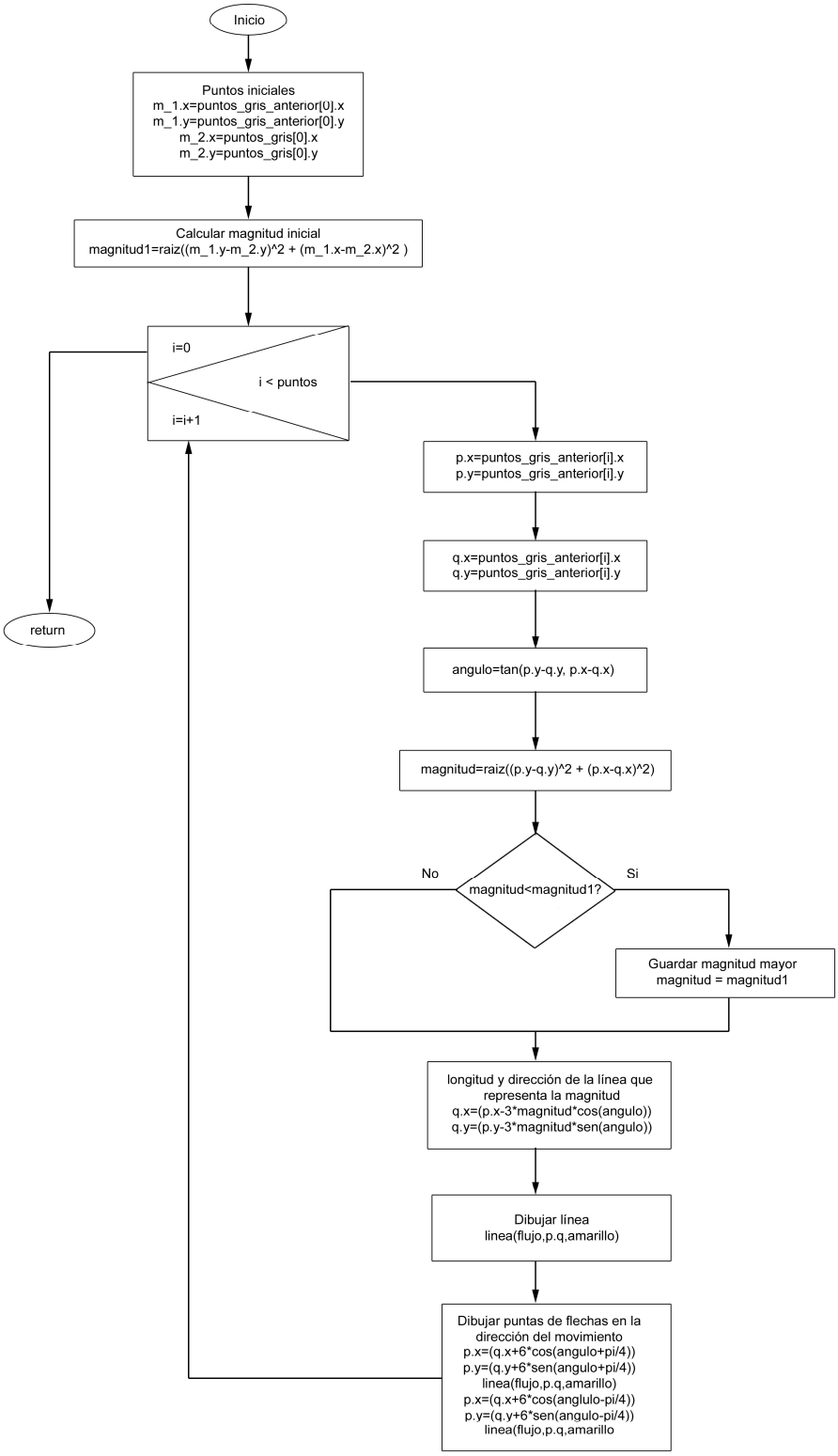


Figura 4.11. Diagrama de flujo para generar el campo de flujo óptico.

4.7 Conclusiones de la etapa de desarrollo del Sistema para Cálculo de Flujo Óptico

El sistema para el cálculo de flujo óptico en una secuencia de imágenes en tiempo real se desarrolla en base a los resultados de las etapas previas. Los resultados en esta última etapa son óptimos gracias a la etapa de modelado y simulación, en la cual se crea una estructura modular en base en las fases para cálculo de flujo óptico por gradientes, propuestas en esta tesis. Al igual que las fases establecidas por Horn- Schunck (1981), estas fases proporcionan un procedimiento general aplicable a los algoritmos para cálculo de flujo óptico por gradientes. Esta estructura básica permite seleccionar algoritmos para la mejora del procesamiento tales como el algoritmo para el seguimiento de puntos característicos de Shi y Tomasi, el cual resuelve algunos de los limitantes de la ecuación restringida de flujo óptico, como la oclusión y la eliminación de puntos que provocan detección de flujo óptico falso. Además se agregó el método piramidal al algoritmo de Lucas-Kanade para el cálculo de flujo óptico, reduciendo a un más el costo computacional. Por último, se diseña un algoritmo para generar el campo de flujo óptico, el cual se puede considerar como una de las aportaciones a este trabajo.

CAPÍTULO V

RESULTADOS Y CONCLUSIONES

Las etapas en las que se desarrolla el proyecto se evalúan en los capítulos previos. La evaluación de cada etapa contribuye al desarrollo de la siguiente, hasta obtener el Sistema para Cálculo de Flujo Óptico a partir de una secuencia de imágenes, en tiempo real. A continuación se describen los resultados de cada etapa, así como los resultados en aplicaciones específicas del proyecto.

5.1 Resultados de la etapa de análisis y prueba

Con los resultados cualitativos de la etapa de análisis y prueba se concluye que el procesamiento de una escena dinámica en cuanto a detección de movimiento tiene una mayor definición con el algoritmo de Lucas-Kanade. En este algoritmo, la detección y la orientación de los vectores de flujo óptico, es más concreta que con el algoritmo de Horn-Schunck, debido a que el cálculo de vectores se realiza solamente en el centro de cada sección en la que se dividió la imagen, lo cual permite la reducción de cálculos, agilizando el proceso de análisis. En cambio, con el algoritmo de Horn-Schunck, los cálculos son iterativos, píxel por píxel. Esto ocasiona que el tiempo del proceso de análisis, con la herramienta de software MATLAB versión 6, fuese 30% mayor. Por esta razón solo se cálculo el flujo óptico en los bordes de los objetos, sin definir magnitud y dirección del movimiento.

5.2 Resultados de la etapa de modelado del sistema

En la segunda etapa, se diseña un modelo a bloques del sistema de cálculo de flujo óptico y detección de movimiento, para su funcionamiento en tiempo real. La finalidad del modelo es comprobar el desempeño de cada algoritmo en cuanto a sensibilidad al ruido, velocidad de procesamiento y cantidad de detecciones adecuadas de regiones en

movimiento. El resultado con el algoritmo de Horn-Schunck, es una mayor afección al ruido producido por el sensor de captura, aún utilizando un filtro previo. El problema del ruido, también ocasiona la detección de regiones en movimiento consideradas como falsas. Estas regiones falsas, se inducen en ciertos casos por la iluminación de la escena. Por otra parte, con el algoritmo de Lucas-Kanade, la afección de ruidos y detección de regiones en movimiento falsas es menor, debido a que este algoritmo aplica un umbral para la reducción del nivel de ruidos y eliminación de movimientos pequeños entre cada cuadro de imagen que evita el cálculo de vectores de flujo óptico falsos. Este umbral se ajusta según las condiciones de la escena. En base a los resultados en esta segunda etapa, se decide utilizar el algoritmo de Lucas-Kanade, en el desarrollo de la etapa final.

5.3 Resultados de la etapa de desarrollo del sistema

Para el desarrollo del sistema y en base a los resultados que arrojan las etapas tanto de análisis y prueba como de modelado, se utiliza el algoritmo de Lucas-Kanade. El rendimiento de este algoritmo es incrementado, agregando un algoritmo previo para la detección y seguimiento de puntos de característicos dentro de las ROI, además de un procedimiento piramidal que facilita el cálculo de flujo óptico. Se obtiene como resultado, un programa funcional donde se implementa como base el algoritmo de Lucas-Kanade en su versión piramidal (Bouquet, 2000) con aportaciones en cuanto a filtrado, localización de puntos característicos y umbralización que permitieron una mejor adaptación a diversas escenas y distintas aplicaciones. El programa detecta el movimiento de los objetos que se desplazan frente a la cámara fija (ver Figura 5.1), el movimiento cuando la cámara se desplaza frente a un escenario fijo y el movimiento cuando tanto la cámara como el escenario se desplazan (ver Figura 5.2). Este movimiento lo define la magnitud y dirección de los vectores de flujo óptico calculados, los cuales se observan gráficamente durante la ejecución del programa (ver Figuras 5.1 y 5.2, flechas amarillas).

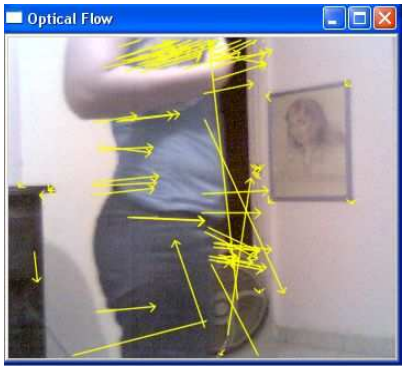


Figura 5.1. Detección de movimiento cuando un objeto se desplaza frente a una cámara fija

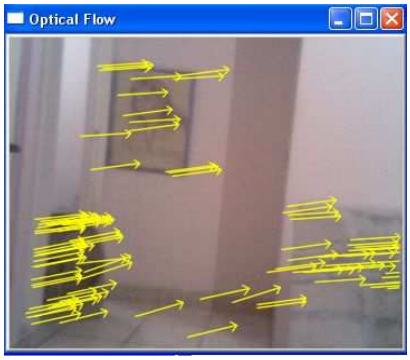


Figura 5.2. Detección de movimiento cuando la cámara se desplaza frente a una escena fija

Con el campo vectorial de flujo óptico obtenido, se definen algunas acciones de movimiento:

- La dirección de desplazamiento de un objeto en movimiento y su velocidad. Este movimiento es lateral, vertical o giratorio; los vectores definen el sentido.

- La dirección de desplazamiento de la cámara en movimiento y su velocidad. Este tipo de movimiento es equivalente a un observador que se desplaza y percibe el campo de flujo óptico. El campo que percibe el observador se calcula y se muestra por medio del sistema. El campo de flujo óptico del movimiento aparente de los objetos, siempre es contrario al desplazamiento del observador (desplazamiento de la cámara).

- El campo vectorial reflectado cuando el observador se aproxima a un objeto, se define cuando los vectores se dispersan a partir de un punto central. Con este campo vectorial detectado, se predice el acercamiento del observador hacia un objeto fijo.

5.4 Resultados obtenidos de la utilidad del sistema en aplicaciones específicas

Se realizan dos aplicaciones específicas, utilizando el sistema como etapa previa de detección de movimiento. La finalidad de estas aplicaciones es comprobar la funcionalidad y portabilidad del sistema como etapa de proceso en aplicaciones que requieren de la detección de movimiento.

5.4.1 Seguimiento de rostros

Se desarrolla una aplicación de seguimiento de patrones, en la que el rostro de una persona es detectado. Una vez detectado, el área del rostro se convierte en la ROI a seguir, mediante la etapa de detección de movimiento. Actualmente, la detección de rostros está incorporada a cámaras de fotografía digital. La diferencia con la aplicación de las cámaras digitales y este desarrollo, es que en las cámaras digitales, no se realiza un seguimiento de rostro, sino una detección del mismo cada determinado tiempo, por lo que si el movimiento de la cámara es brusco, se pierde dicha detección. En la aplicación desarrollada en este trabajo, se utiliza la etapa de detección de movimiento, para el seguimiento continuo de una región determinada de tal modo que el seguimiento del rostro no se pierde. Los resultados obtenidos de esta aplicación fueron satisfactorios. La velocidad de procesamiento alcanzada en ambas aplicaciones fue de 30 cuadros por segundo (ver Figura 5.3).

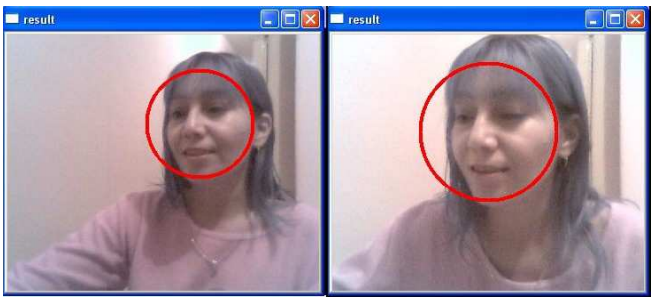


Figura 5.3. Reconocimiento de rostros y seguimiento mediante flujo óptico

5.4.2 Instrumento para análisis cuantitativo del movimiento de microestructuras

Debido al comportamiento observado de los vectores del campo del flujo óptico, se decide aplicar el programa como un instrumento para análisis cuantitativo del movimiento de microestructuras (vea Figura 5.4), para la determinación de planos focales en un microscopio. El resultado con esta aplicación es el enfoque del lente, el cual coincide con la estabilización de los vectores de flujo óptico (vea Figura 5.5), logrando con ello el control del movimiento axial y lateral de planos en microestructuras observadas en un microscopio.

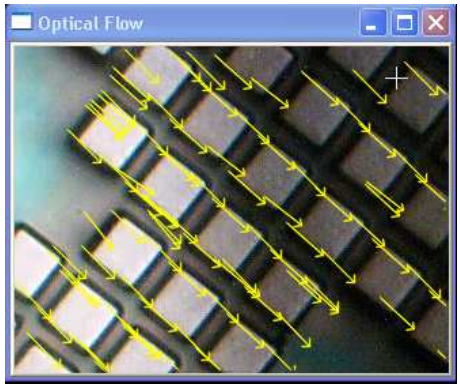


Figura 5.4. Movimiento lateral de microestructuras captado por los vectores de flujo óptico

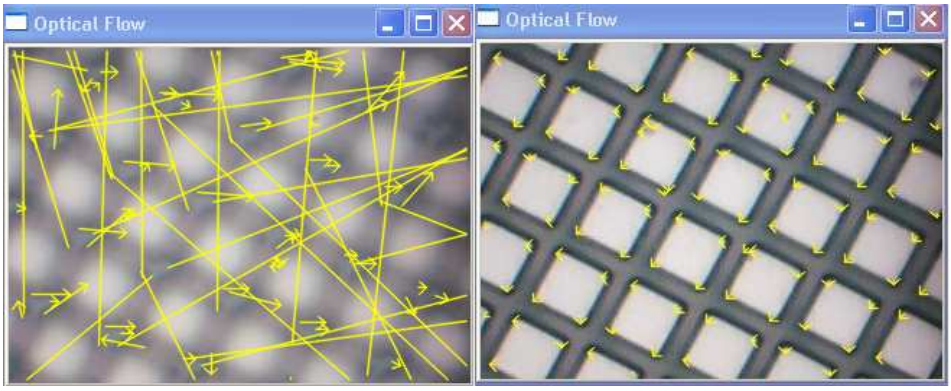


Figura 5.5. Movimiento axial de planos de microestructuras captado por los vectores de flujo óptico

5.4.3 Instrumento de evaluación de algoritmos de flujo óptico

La implementación de los algoritmos por software permite la evaluación de la efectividad de los mismos de una manera más simple y rápida, ya que las modificaciones para mejorar su rendimiento implican un costo mínimo de tiempo. La implementación de algoritmos por software también permite desarrollar instrumentos o herramientas que facilitan el análisis y pruebas de los algoritmos de acuerdo a las aplicaciones que se desean desarrollar, debido al diseño de funciones portables y configurables donde se integran estos algoritmos.

5.5 Discusión

En este trabajo se define como objetivo principal, la implementación por software de algoritmos para el cálculo de flujo óptico a partir de una secuencia de imágenes procesada en tiempo real, en consideración a investigaciones anteriores donde implementación por software logra el análisis de secuencias de imágenes consistentes en dos o tres cuadros de imagen contenidas en archivos (Horn y Schunck, 1981; Lucas y Kandade 1981; Kusina, 2000; Backer *et al.*2003). También se considera el hecho de investigaciones que realizan la implementación en tiempo real, pero en aplicaciones específicas y por medio de hardware. El objetivo definido en esta investigación, es alcanzado en su totalidad, ya que se consigue el análisis en tiempo real de una secuencia de imágenes en la que se determina el campo de flujo óptico por medio de algoritmos implementados por software.

Uno de los objetivos específicos definidos es utilización del sistema en aplicaciones específicas que requieren de la detección de movimiento por medio de los vectores de flujo óptico. En este objetivo particular, se obtienen resultados satisfactorios en dos aplicaciones de detección de movimiento en tiempo real. La primera es el seguimiento de patrones, donde el área del rostro, se convierte en la ROI de detección de movimiento. La segunda es la determinación de planos focales de un microscopio. Cabe resaltar que en esta última aplicación, aún no se ha experimentado con flujo óptico para lograr tal objetivo. Estos resultados se consiguen con herramientas simples tales como una computadora de bajo rendimiento y una cámara Web.

5.6 Recomendaciones

Se sugiere la mejora del sistema utilizando dispositivos de hardware eficientes tales como una cámara de mayor resolución y de menor susceptibilidad al ruido, así como un equipo de cómputo de alto rendimiento. Se plantea la adaptación del sistema, como una herramienta de evaluación y simulación de aplicaciones que implican la detección de movimiento.

5.7 Aportaciones

Como se ha mencionado en reiteradas ocasiones, el sistema debe calcular el flujo óptico en tiempo real, a una velocidad adecuada (la máxima velocidad especificada para el sensor de video utilizado), mediante procesamiento por software. Este objetivo se consigue alcanzando la velocidad máxima de captura de la cámara utilizada, la cual es de 30 cuadros por segundo. También se optimiza el algoritmo de Lucas-Kanade, utilizando técnicas para el seguimiento de puntos característicos y el procesamiento piramidal de las imágenes. Además, en la implementación del sistema se plantea un cambio en las fases para el cálculo de flujo óptico por gradientes que permite intercambiar diferentes algoritmos en cada fase sin la alteración del sistema. Otra de las aportaciones importantes es el algoritmo diseñado para la generación del campo de flujo óptico el cual se observa gráficamente de manera simultánea sobre la secuencia de video. Por último, una aportación imprevista es la utilización del campo de flujo óptico para la determinación de planos focales de un microscopio. Esta aplicación no ha sido abordada aún por otros investigadores.

5.8 Conclusiones

La implementación por software de algoritmos para cálculo de flujo óptico, brinda notables ventajas sobre las implementaciones por hardware. En la implementación por software, la evaluación de la eficiencia de los algoritmos para cálculo de flujo óptico es más simple, porque su diseño es modular y general. Debido a que las implementaciones

de algoritmos por hardware se diseñan para una aplicación en particular, solo se evalúa el desempeño del algoritmo ante la solución específica del problema. Las implementaciones por software, utilizan funciones modulares, donde la variación de parámetros, permite la optimización de procedimientos de cálculo, los cuales se ajustan a las condiciones de determinada aplicación. Por ejemplo, para la detección del movimiento en una secuencia de video, se puede variar los rangos de niveles de gris, dependiendo de la iluminación de la escena. Además, resulta práctico que los programas diseñados se utilicen en equipos de cómputo propósito general, permitiendo el desarrollo de software transportable y de bajo costo, cuya utilización va en aumento en este tipo de aplicaciones.

REFERENCIAS

Baker, S., Gross, R., Matthews, I., e Ishikawa, T. (2003). “Lucas-Kanade 20 Years On: A Unifying Framework” report CMU-RI-TR-03-01, Robotics Institute, Carnegie Mellon University.

Ballard, D. and Brown, C. (1982). "Computer vision", Prentice-Hall, New Jersey.

Barron J.L. and Beuachemin S. S. (1995). “The computations of Optical Flow”. ACM Computing Surveys, vol.27, no. 3, pp. 433-467.

Bouquet, J. (2000). “Pyramidal Impremention of the Lucas Kanade Feature Tracker Description of the Algorithm”, Intel Corporation, Microprocessor Research Labs.

Cobos Arribas, P. (2001). “Arquitectura Hardware para la Extracción de Invariantes Ópticos, a partir del flujo óptico, en tiempo real”. Tesis Doctoral en Tecnologías Especiales, Universidad politécnica de Madrid, Madrid, España.

Gibson, J. (1979). “The ecological approach to visual perception”. Boston_ Hougton Mifflin.

Gu, H., Shirai, Y., y Asada, M. (1996). “MDL-based segmentation and motion modeling in a long image sequence scene with multiple independently moving objects”. IEEE Transitions on pattern analysis and machine intelligence, vol. 18, no. 1, (pp. 59-56).

Gupta, N. and Kanal, L. (1995). “3-D motion estimation from motion field”. Artificial intelligence, no. 11, (pp. 45-86).

Gupta, S. and Sproull, R. (1981). "Filtering Edges for Gray-Scale Displays", Computer Graphics, vol. 15, no. 3.

Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In Alvey Vision Conference, pp. 147-151.

Hartley, R. (1985) “A Gaussian-Weighted Multi-Resolution Edge Detector”, Computer Vision, Graphics, and Image Processing. Vol. 30, No. 1, pp. 70-83.

Horn, B. and Schunk, B. (1981). “Determining optical Flow”. Artificial Intelligence, no. 17, (pp. 185-203).

Jain R., Kasturi R., and Schunck, B.G. (1995). “MachineVision”. Munson E.M. (Ed.), Singapore: McGraw-Hill.

Kottke, D. and Sun, Y. (1994). “Motion estimation via cluster matching.” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 16, no. 11, pp. 1128-1132.

Kuzina, T.Y. (2000). “Investigación e implementación de los algoritmos del campo de visión dinámica por computadora”. Departamento de Ingeniería en Sistemas Computacionales. Universidad de las Ameritas, Puebla. [Documento www]. Recuperado: http://www.pue.udlap.mx/~tesis/kusina_ty/resumen.pdf.

Laplante, P.A. and Stoyenko, A.D. (1996). “Real time imaging. Theory, techniques, and applications. New York: IEEE Press.

Lay D.C. (2001). “Algebra Lineal y sus Aplicaciones” (2da. ed. actualizada), Prentice-Hall, México.

Low, A. (1991). “Computer Vision and Imaging Processing, McGraw-Hill, New York.

Lucas, B.D. and Kanade, T. (1981). “An Iterative Image Registration Technique with an Application to Stereo Vision”, Proc 7th Intl Joint Con ton Artificial Intelligence (IJCAI) 1981, August 24-28 Vancouver, British Columbia, pp. 674-679.

Martinez, R. (2002). “Ciencia y representación del ojo”. Facultad de Ciencias, Universidad Autónoma de México. Recuperado: http://www.ejournal.unam.mx/ciencias/no66/CNS06606.pdf

Nakamura, S. (1997). “Analisis numérico y visualización gráfica con MATLAB”. (Trad. de Escalona García R.). México: Prentice Hall.

Nuño, M., Arias, E. y Feregrino, U. (2005). “Implementación Hardware de Aplicaciones de la Pirámide”. Instituto Nacional de Astrofísica, Óptica y Electrónica. Cholula, Puebla; México.

OpenCV, Intel Open Source Computes Vision Library. (2000-2006). “Biblioteca de funciones para procesamiento de imágenes”. http://www.sourceforge.net/projects/opencvlibrary.

Otte, M and Nangel, H. (1995). “Estimation of optical flow based on higher-order spatiotemporal derivatives in interlaced and non-interlaced images sequences”. Artificial intelligence 78, (pp. 5-43).

Pajares, M. G. (2004). “Imágenes digitales: procesamiento práctico con Java”. (1ra ed.) Editorial Alfaomega.

Pajares, M. G. De la Cruz, J.M. (2002). “Visión por computador: Imágenes digitales y aplicaciones”. Madrid España. Editorial Alfaomega, Ra-Ma.

Panjwani, D. and Healey, G. (1995). “Markov random field models for unsupervised segmentation of textured color images”. IEEE Transactions on pattern analysis and machine intelligence, vol. 17, no. 10, pp. 939-954.

Pardàs, M. and Salembier, P. (1994). “3D morphological segmentation and motion estimation from image secuences”. Signal Processing, vol. 38, no. 1, pp. 31-43.

Sandhana, L. (2004). “Bugs Taking Over Robot Guidance”. Wired Magazine, Science topic. Recuperado: http://www.wired.com/science/discoveries/news/2004/01/61883?currentPage=all

Schweitzer, H. (1995). “Occam algorithms for computing visual motion”. IEEE Transactions on pattern analysis and machine intelligence, vol. 17, no.11, (pp. 1033-1042).

Shi, J. and Tomasi, C. (1994). “Good Features to Track” IEEE Conference on Computer Vision and Pattern Recognition (CVPR94) Seattle, June 1994.

Smith, S.M., and Brady, J.M. (1997). “SUSAN - a new approach to low level image processing”. Int. Journal of Computer Vision, vol 23, no. 1, pp: 45-78.

Soille, P. ¨Morphological Image Analysis¨. 2nd ed. New York, Springer, 2003.

The MathWorks, Inc. (2004-2007). “User’s Guide”. Natick, Massachussets : Autor.

Uchiyama, T. and Arbib, M. (1994). “Color image segmentation using competitive learning”. IEEE Transactions on pattern analysis and machine intelligence, vol. 16, no. 12, pp. 1197-1206.

Verri, A. and Poggio, T. (1989). “Motion field and optical flow: Qualitative properties”. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 11, no.5, pp: 490-498.

Zhang, Z. (1995). “Estimating motion and structure from correspondence of line segments between two perspective images”. IEEE Transactions on pattern analysis and machine inteligence, vol. 17, no.12, pp. 1129-1139.

Zuloaga, A., Martín, J. L. y Ezquerra, J. A. (1998). “Hardware Architecture for Optical Flow Estimation in Real Time”, Proc of. International Conference on Image Processing, vol. 3, pp. 972-976.

Zuloaga, I. A. (1998). “Visión artificial dinámica, determinación del movimiento a partir de una secuencia de imágenes”. Bilbao, España. [Documento www]. Recuperado: http://www.geocities.com/aiztol.geo/docu003.pdf

Yamamoto, M. (1989) “A general aperture problem for direct estimation of 3-D motion parameters.” IEEE Transactions on pattern analysis and machine inteligence, vol. 11, no. 5, pp. 528-536.

ANEXO A

PROGRAMAS DE PRUEBA APLICANDO EL ALGORITMO DE HORN-SCHUNCK.

Detección de desplazamiento

Programa para detectar desplazamiento de un objeto de un cuadro en t_0 a un cuadro en t_1. Los resultados obtenidos de estos programas fueron explicados en el capitulo IV en la etapa de análisis y prueba.

```
I=imread('c:\bolaX.jpg');  %Lectura de la primera imagen
Ia=rgb2gray(I); %Conversión a niveles de gris de la primera imagen
Ix = mat2gray(filter2(fspecial('sobel'),Ia)) %Obtención del gradiente temporal horizontal de la primera imagen.
Iy = mat2gray(filter2(fspecial('sobel')',Ia)) %Obtención del gradiente temporal vertical de la primera imagen.
figure,imshow(Ix)  %Mostrar gradiente en X.
figure,imshow(Iy)   %Mostrar gradiente en Y.

I4=imread('c:\bolaY.jpg') %Lectura de la segunda imagen.
Ib=rgb2gray(I4); %Conversión a niveles de gris de la segunda imagen.
Iw = mat2gray(filter2(fspecial('sobel'),Ib)) %Obtención del gradiente temporal horizontal de la segunda imagen.
Iz = mat2gray(filter2(fspecial('sobel')',Ib)) %Obtención del gradiente temporal vertical de la segunda imagen.
figure,imshow(Iw) %Mostrar gradiente en X.
figure,imshow(Iz)  %Mostrar gradiente en Y.

I7=imsubtract(Iw,Ix)  %Diferencia de gradientes en X (Gradiente temporal en X).
```

figure,imshow(I7) %Muestra diferencia.

I8=imadjust(I7,[0 20/255],[1 0]); %Ajuste del resultado a un rango determinado para ser considerado como borde (Aplicación de Umbral).

figure,imshow(I8) %Muestra el resultado de la umbralización en X.

I9=imsubtract(Iz,Iy) %Diferencia de gradientes en Y (Gradiente temporal en Y).

figure,imshow(I9) %Muestra diferencia.

I10=imadjust(I9,[0 20/255],[1 0]); %Aplicación de Umbral.

figure,imshow(I10) %Muestra el resultado de la umbralización en Y.

I11=imadd(I9,I7) % Suma de gradientes temporales.

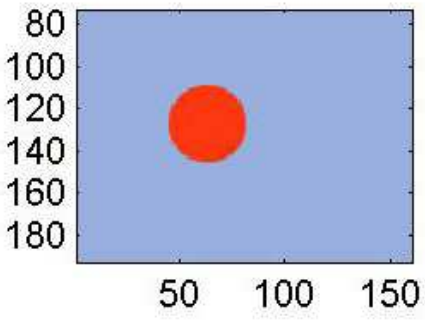
figure,imshow(I11) %Muestra total de gradientes temporales.

I12=imadjust(I11,[0 20/255],[1 0]); %Aplicación de umbralización para ajustes de bordes.

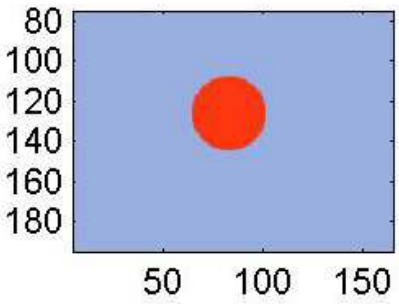
figure,imshow(I12) %Muestra resultado de la umbralización anterior.

%Despliegue de gráficas resultantes (ver fig. 3.1.1.4.)

subplot(4,2,1), subimage(I)

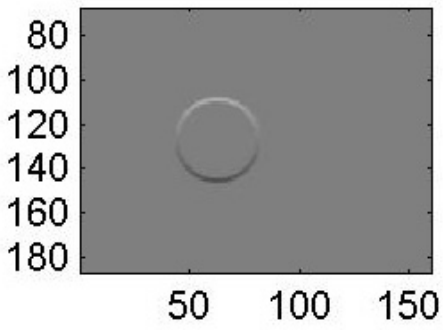


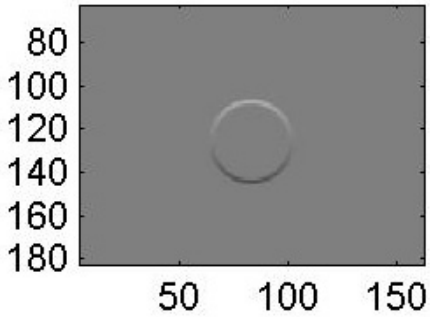
subplot(4,2,2), subimage(I4)



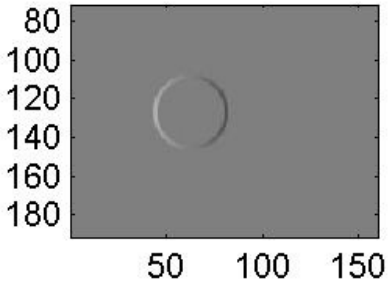
subplot(4,2,3), subimage(Ix)

subplot(4,2,4), subimage(Iz)

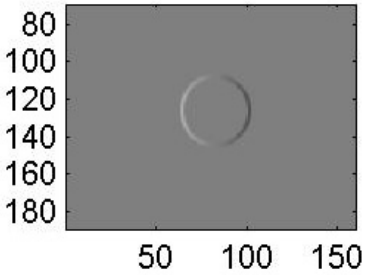




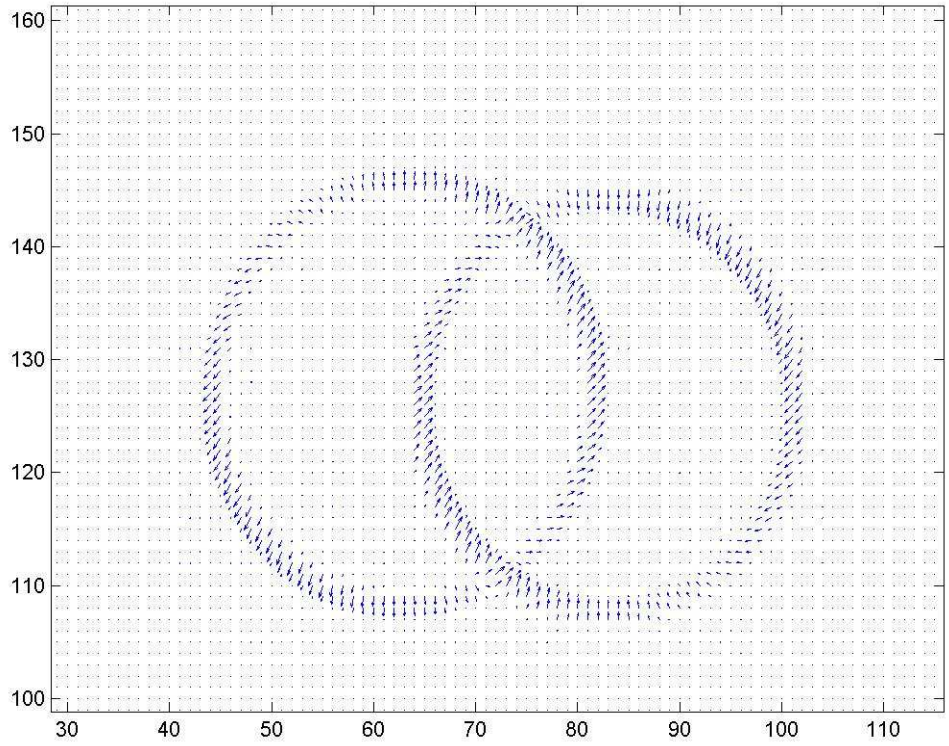
subplot(4,2,5), subimage(Iy)



subplot(4,2,6), subimage(Iw)




```
Vx=im2double(I9);   %Cálculo del vector de flujo óptico en X.
Vy=im2double(I11); %Cálculo del vector de flujo óptico en Y

quiver(Vx,Vy)        %Función que despliega los vectores del flujo óptico.
```

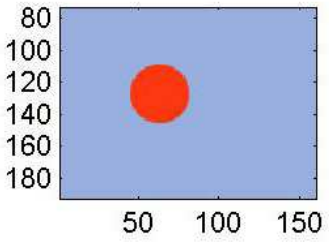


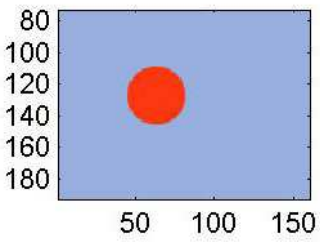
Comprobación de la ausencia de movimiento entre dos cuadros de una escena

En este programa se analizaron dos cuadros de imagen donde un objeto permanece en la misma posición en ambos cuadros. El resultado obtenido fue la detección de ausencia de movimiento mediante el cálculo de gradientes temporales que en este caso es igual a cero, puesto que los gradientes temporales se obtienen de la diferencia de gradientes espaciales entre t_0 y t_1. En ambos tiempos el valor de los gradientes espaciales es el mismo, por lo tanto su diferencia es de cero.

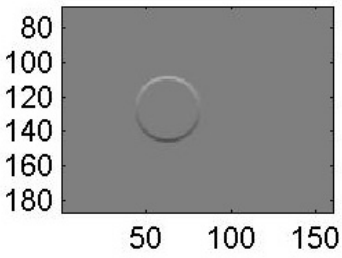
```
I=imread('c:\bolaX.jpg'); %Lee imagen con un objeto en determinada posición.
Ia=rgb2gray(I);
Iy = mat2gray(filter2(fspecial('sobel'),Ia))
Ix = mat2gray(filter2(fspecial('sobel')',Ia))
```

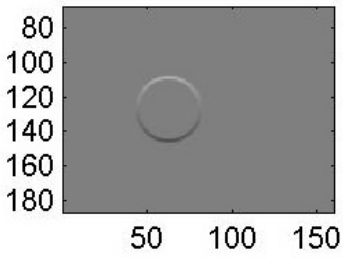
```
figure,imshow(Iy)
figure,imshow(Ix)

I4=imread('c:\bolaX.jpg'); %Carga la imagen anterior con el objeto en la misma posición.
Ib=rgb2gray(I4);
Iw = mat2gray(filter2(fspecial('sobel'),Ib))
Iz = mat2gray(filter2(fspecial('sobel')',Ib))
figure,imshow(Iw)
figure,imshow(Iz)

I7=imsubtract(Iw,Iy)    % Sustracción de los gradientes horizontales
figure,imshow(I7)

I9=imsubtract(Iz,Ix) % Sustracción de los gradientes verticales.
figure,imshow(I9)
I11=imadd(I9,I7)
figure,imshow(I11)

%Despliegue de gráficas (ver fig. 3.1.1.5)
subplot(4,2,1), subimage(I)          subplot(4,2,2), subimage(I4)
```



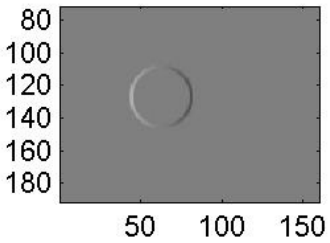


subplot(4,2,3), subimage(Iy)   subplot(4,2,4), subimage(Iw)

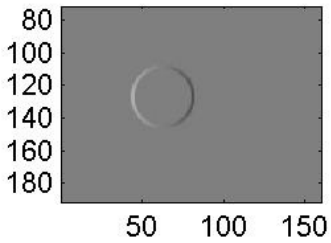




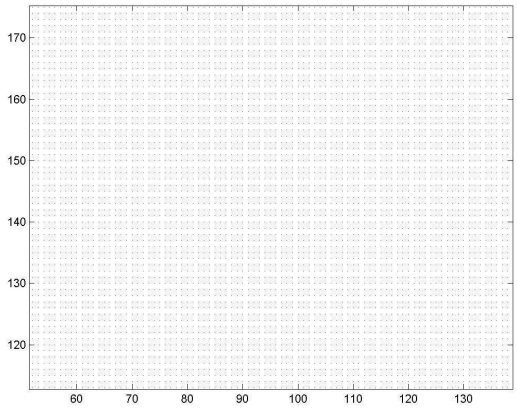
subplot(4,2,5), subimage(Ix)



subplot(4,2,6), subimage(Iz)




```
Vx=im2double(I9);
Vy=im2double(I11);
quiver(Vx,Vy) %Despliegue de vectores del flujo óptico.
```

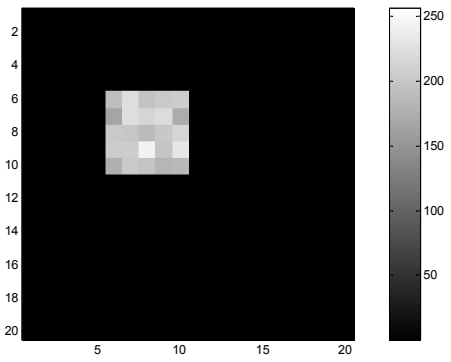


ANEXO B

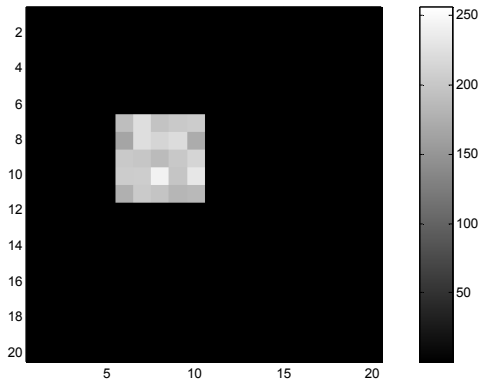
PROGRAMAS DE PRUEBA APLICANDO EL ALGORITMO DE LUCAS-KANADE.

El objetivo de este programa es analizar el método de estimación del movimiento basado en la utilización de la ecuación del flujo óptico por Lucas-Kanade, al igual que con el algoritmo de Horn-Schunck. Por lo tanto, dados dos cuadros de una secuencia de imágenes, el programa calcula, cuál ha sido el desplazamiento de los objetos de un cuadro al otro.

```
% Tamaño de imagen
nf=20; % Columnas
nc=20; % Renglones

% Imagen sintética en t=0
A0=zeros(nf,nc);  % Inizalización
Region=200+ 20*randn(5,5); %Generación de valores de gris aleatorios
A0(6:10,6:10)=Region; % asignación de valores y de posición de la imagen

% Imagen a buscar (t=1)
A1=zeros(nf,nc); % Inizalización
A1(7:11,6:10)=Region; % asignación de valores y de posición de la imagen
% Muestra de imágenes en pantalla
image(A0);
colormap(gray(256)), axis equal ij tight, colorbar;
```



```
figure, image(A1);
colormap(gray(256)), axis equal ij tight, colorbar;
```



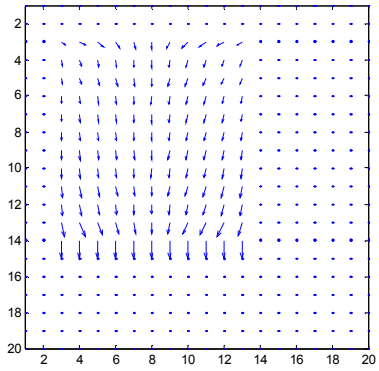

```
% Se fija el tamaño de bloque en 5x5
bx=5;
by=5;

% Filtro de convolución para ambas imágenes
A0=conv2(A0,ones(3)/9,'same');
A1=conv2(A1,ones(3)/9,'same');

% Cálculo de las derivadas
Hx=[-1 0 1]';
Hy=[-1 0 1];

Ix=imfilter(A0,Hx,'symmetric');
Iy=imfilter(A0,Hy,'symmetric');

%Cálculo del gradiente temporal
It=(A1-A0);

%Cálculo de las matrices requeridas para la ecuación 3.1.2.2., descrita en el capítulo IV.
mascarasuma = ones(bx,by); % Matriz auxiliar inicializada en valores de 1
Ix2 = Ix.^2;
```

```
Suma_Ix2 = imfilter(Ix2,mascarasuma,'symmetric');
Iy2 = Iy.^2;
Suma_Iy2 = imfilter(Iy2,mascarasuma,'symmetric');
IxIy = Ix.*Iy;
Suma_IxIy = imfilter(IxIy,mascarasuma,'symmetric');
IxIt=Ix.*It;
Suma_IxIt = imfilter(IxIt,mascarasuma,'symmetric');
IyIt=Iy.*It;
Suma_IyIt = imfilter(IyIt,mascarasuma,'symmetric');

clear Hx Hy Ix Iy It Ix2 Iy2 IxIy IxIt IyIt mascarasuma

%Inicialización de los vectores horizontal y vertical para el cálculo del flujo óptico
U = zeros(nf, nc);
V = U;

%Solución de la ecuación 3.1.2.2.
for i = 1:nf
  for j = 1:nc

%Construcción de la matriz C con las sumatorias de la ecuación 3.1.2.2.
    C=([Suma_Ix2(i,j) Suma_IxIy(i,j);Suma_IxIy(i,j) Suma_Iy2(i,j)]);
    %Construcción de la matriz b com las sumatorias de los vectores temporales
      en X y Y.
    b=-[Suma_IxIt(i,j); Suma_IyIt(i,j)];

c=(pinv(C)*b); %Calcula el inverso de la matriz C
    U(i,j)=c(1); %Flujo Horizontal
    V(i,j)=c(2); %Flujo vertical
  end;
end;
```

figure;

quiver(V,U,0); % Despliega la velocidad de los vectores de flujo óptico en forma de flechas que indican la orientación del movimiento.

axis equal ij tight



% Se compara la imagen A1 con su predicción.

% Para ello primero se desplaza A0 con los vectores de desplazamiento

% obtenidos

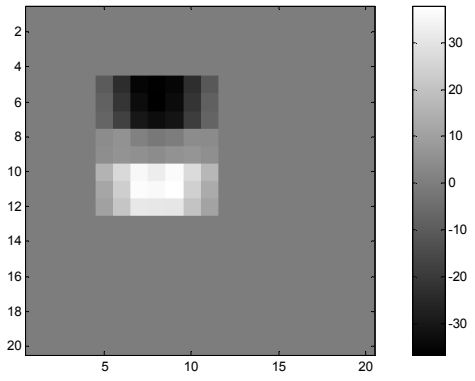
[cols,filas]=meshgrid(1:nc,1:nf);

A0w=griddata(cols+V,filas+U,A0,cols,filas);

% Se dibuja las diferencias

figure, imagesc(A1-A0w);

colormap(gray(256)), axis equal ij tight, colorbar;



ANEXO C

PRODUCTOS GENERADOS DEL PROYECTO DE INVESTIGACIÓN

1. Valenzuela, M., Guerrero, J., Bravo, M, y Reyna, M. (2007, Septiembre 27). "Implementación de Algoritmos para Cálculo de Flujo Óptico en una Secuencia de Imágenes". Artículo publicado en el Primer Coloquio de Ciencias de la Computación. Morelia, Michoacán; México. http://enc.sncc.org.mx.

2. Valenzuela, M. (2007, Octubre). "Visión Artificial". Conferencia impartida en el VII Simposio Internaconal de Ingeniería, DECIVEL 07, Proyectando el Ingenio. Universidad Autónoma de Baja California. Mexicali, B.C.; México.

3. Valenzuela, M. Guerrero, J y Martínez, M. (2007, Septiembre 15). "Sistema de Visión por Computadora". Segundo Concurso Estatal de Creatividad Científica y de Procesos. Segundo Lugar obtenido en proyectos de investigación. Universidad Autónoma de Baja California. Ensenada, B.C.; México.

4. Valenzuela, M. (2007, Noviembre 16). "Implementación de Algoritmos para Cálculo de Flujo Óptico en una Secuencia de Imágenes". Conferencia impartida en el IV Semana de LASCA. Centro de Estudios Superiores del Estado de Sonora. San Luis R.C.; México.

5. Valenzuela, M. y Bravo, M. (2007, Diciembre). "Implementación de Algoritmos para Cálculo de Flujo Óptico en una Secuencia de Imágenes". Artículo publicado en el Segundo Coloquio de Posgrado del Programa de Maestría y Doctorado en Ciencias e Ingeniería. Intituto de Ingeniería. Universidad Autónoma de Baja California. Mexicali, B.C.; México.

GLOSARIO

A continuación se definen algunos términos y conceptos utilizados en este trabajo.

Arcivo de tipo AVI (Audio Video Interleaved): es un formato para archivos que contienen audio y video intercalado propio de Windows, que sólo se puede ejecutar bajo esta plataforma.

Campo de flujo óptico: Es el campo de velocidad que representa el movimiento tridimensional de puntos de los objetos a través del movimiento bidimensional de la imagen (Pajares y de la Cruz, 2002).

Campo de movimiento *(Motion Field):* Es un arreglo bidimensional de vectores que representan la velocidad de cada punto de una imagen, o lo que es lo mismo, es la proyección en perspectiva del campo tridimensional de vectores de velocidad de una escena en movimiento en el plano de la imagen (Verry y Poggio, 1989). El campo de movimiento es un concepto abstracto y no puede ser recuperado de una secuencia de imágenes. Lo que sí puede ser recuperado de una secuencia de imágenes es el flujo óptico (Gupta y Kanal, 1995).

Campo de visión: Es el lugar en el que un sensor de imagen puede captar la información lumínica (Pajares y de la Cruz, 2002).

Cuadro *(Frame)*: Es una matriz bidimensional de dos valores de intensidad lumínica obtenidos para un tiempo t_0 constante $I\,[x, y, t_0]$ (Zuloaga, 1998).

DirectX: Es una colección de APIs creadas y recreadas para facilitar las tareas complejas relacionadas con la programación en la plataforma Microsoft Windows.

Flujo óptico *(Optical flow):* Es la velocidad aparente de las estructuras de niveles de gris, representado por un arreglo bidimensional de valores (Otte y Nagel, 1995). El flujo óptico puede considerarse como una aproximación del campo de movimiento, pero a diferencia de aquel puede ser obtenido de una secuencia de imágenes (Gupta y Kanal, 1995). Las variaciones en los niveles de gris pueden ser debidas al movimiento de los objetos en las secuencias de imágenes, por ello a partir del flujo óptico es posible determinar el movimiento de los objetos en secuencias de imágenes (Horn y Schunck, 1981).

Imagen *(Image)*: Es la proyección en perspectiva en el plano bidimensional de una escena tridimensional en un determinado instante de tiempo t_0. (Pajares, 2004 y Zuloaga, 1998).

Imagen a color *(Color Image)*: Este tipo de imagen es más completa pues es posible aplicarle todo tipo de procesamiento y se le puede convertir fácilmente en una imagen de tonos de grises o binaria (Pajares, 2004).

Imagen binaria *(Binary Image)*: Es una imagen compuesta sólo de dos colores generalmente en blanco y negro. Son frecuentemente utilizadas en el procesamiento digital ya que permite identificar rápidamente los objetos de interés en una imagen y aplicarles diversos algoritmos (Pajares, 2004).

Imagen de tono de grises *(Gray Image)*: Conocida también como imagen a blanco y negro (Black and White, BW) aunque en realidad está compuesta de tonos de grises. Las más común es la compuesta de 256 tonos de grises donde un píxel de valor 0 es negro y un píxel de valor 255 es blanco (máxima intensidad) (Pajares, 2004).

Imagen real: Se entiende por imagen real, la fotografía tomada de una escena, convertida a un archivo de tipo imagen.

Imagen sintética: Se entiende por imagen sintética aquella que es creada mediante algún tipo de software, con ciertas características para la realización de pruebas.

Invariantes ópticos: Medidas que conducen al movimiento y no varían ante los cambios en los parámetros de la imagen como la luminosidad, la escala o la rotación (Cobos, 2001).

Métodos de los gradientes: Consiste en determinar los cambios (gradientes) espaciales y temporales del patrón de grises de la imagen y a partir de ellos obtener el flujo óptico (Zuloaga, 1998).

Operador intervalo de umbral binario: Esta clase de transformación crea una imagen de salida binaria a partir de una imagen de grises, donde todos los valores de gris cuyo valor está en el intervalo definido por p1 y p2 son transformados a 255 y todos los valores fuera de ese intervalo a 0.

$$q = \begin{cases} 255 \; para \; p \le p_1 \quad o \quad p \ge p_2 \\ 0 \qquad para \qquad p_1 < p < p_2 \end{cases} \tag{1.2}$$

Operador umbral: Es una clase de transformación que crea una imagen de salida binaria a partir de una imagen de grises, donde el nivel de transición está dado por el parámetro de entrada p_1.

$$q = \begin{cases} 0 \quad para \; p \le p1 \\ 255 \; para \; p > p1 \end{cases} \tag{1.1}$$

Parámetros del movimiento *(Motion Parameters):* Es aquel conjunto de números los cuales representan el movimiento de un punto o un grupo de puntos en el espacio y el tiempo. Tales coeficientes pueden representar las coordenadas espaciales, la velocidad lineal, la velocidad angular, magnitud, dirección, etc. (Zuloaga, 1998).

Píxel *(Píxel: Picture Element):* Es cada una de las posiciones en que es discretizada una imagen, o lo que es lo mismo, cada una de las posiciones de un cuadro. De esta manera un cuadro es un arreglo bidimensional de píxeles, cada uno con cierto valor lumínico *(*Zuloaga, 1998).

Regla de Cramer: Es un teorema en álgebra lineal, que da la solución de un sistema lineal de ecuaciones en términos de determinantes. Recibe este nombre en honor a Gabriel Cramer (1704 - 1752).

Secuencia digitalizada de vídeo o de imágenes *(Digitized video sequence).* Es una muestra discreta en espacio y tiempo *I [x, y, t]* de una función continua de intensidad lumínica *I(x, y, t)* (Schweitzer, 1995).

Segmentación de las imágenes: Consiste en identificar una serie de características locales (objetos, bloques, líneas etc.) en una imagen, tratar de obtener las correspondencias con la otra imagen y determinar los movimientos (Zuloaga, 1998).

Segmentación en bloques: Modelan las imágenes dividiéndolas en rectángulos de tamaños variables que se adaptan a áreas de nivel de gris o color más o menos homogéneo (Schweitzer, 1995).

Segmentación en línea: Algoritmos de segmentación que extraen el contorno de los objetos, modelan estos como segmentos de línea y determinan los parámetros de movimiento en base al desplazamiento de los segmentos entre las sucesivas imágenes (Gu, Shirai y Asada, 1996).