UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SEGUIMIENTO DE OBJETIVOS USANDO FILTRO KALMAN

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

P R E S E N T A :

ROSALES FLORES SABRINA

QUE PARA OBTENER EL TÍTULO DE

INGENIERO ELÉCTRICO ELECTRÓNICO

P R E S E N T A :

CARRANZA MARTÍNEZ JOSÉ DE JESÚS

DIRECTOR DE TESIS:

M. en I. ALBERTO FUENTES MAYA

MÉXICO, D.F. Junio 2005

SISTEMA DE SEGUIMIENTO DE OBJETIVOS MÓVILES USANDO FILTRO KALMAN

ÍNDICE

Índice de figuras. iv

Índice de tablas. vi

Objetivo. vii

Hipótesis. viii

Introducción. ix

Capítulo I Antecedentes

I.I Sistema de Visión humana. 1

I.II La visión humana y el procesamiento digital de imágenes. 4

I.III Analogía Ojo-Cámara. 7

I.IV Requerimientos de hardware y software para los sistemas de visión por computadora. 8

I.V Cámara de video analógica. 9

I.VI Tarjeta capturadora de imagen. 10

I.VII Cámara CCD. 12

I.VIII Unidad de salida de una cámara de video. 15

I.IX Escaneo progresivo y trenzado. 16

I.X Forma de onda típica de un video analógico. 16

I.XI Tabla de formatos de video analógico. 18

I.XII Comunicación entre dispositivos digitales. 20

I.XIII Comunicación serial. 20

I.XIV Comunicación serial síncrona. 21

I.XV Comunicación serial asíncrona. 22

I.XVI Comunicación paralela. 23

I.XVII Comunicación USB. 26

Capítulo II Conceptos generales del procesamiento de imágenes

II.I Representación digital de imágenes 27

II.II Fases del procesamiento digital de imágenes 28

II.III Modelo de imagen simple. 30

II.IV Muestreo y cuantificación. 31

II.V Muestreo uniforme y cuantificación. 32

II.VI Muestreo no uniforme y cuantificación. 33

II.VII Definición de píxel. 33

II.VIII Relaciones básicas entre píxeles. 34

II.IX Vecinos de un píxel. 34

II.X Conectividad. 35

II.XI Medidas de distancia. 36

II.XII Operaciones aritmético-lógicas. 37

II.XIII Histograma. 38

Capítulo III Procesamiento digital de imágenes

III.I Segmentación de imágenes. 39

III.II Detección de bordes. 40

III.III Detección de discontinuidades. 42

III.IV Enlazado de bordes y detección de límites. 43

III.V Umbralización. 43

III.VI Segmentación orientada a regiones. 45

III.VII Utilización del movimiento en la segmentación. 46

III.VIII Ejemplo de aplicación con procesamiento digital de imagen. 47

Capítulo IV Filtros predictivos

IV.I Filtros predictivos. 53

IV.II Filtros g-h y g-h-k. 54

IV.III Filtros α y β. 62

IV.IV Filtro Kalman. 62

IV.V Filtro Kalman extendido. 69

Capítulo V Desarrollo del sistema de seguimiento de objetivos móviles usando filtro Kalman

V.I Componentes electrónicos y programas necesarios para el desarrollo del sistema de seguimiento de objetivos móviles usando filtro Kalman. 71

V.II Descripción de elementos electrónicos y programas utilizados para el desarrollo del sistema de seguimiento de objetivos móviles usando filtro Kalman. 75

V.III Descripción del proceso de análisis y obtención de resultados del sistema de seguimiento de objetivos móviles usando filtro Kalman. 80

V.IV Captura de imagen del sistema de seguimiento de objetivos móviles usando filtro Kalman. 80

V.V Proceso de elección del objeto a seguir del sistema de seguimiento de objetivos móviles usando filtro Kalman. 81

V.VI Procesamiento digital de imagen del sistema de seguimiento de objetivos móviles usando filtro Kalman. 84

V.VII Filtro Kalman del sistema de seguimiento de objetivos móviles usando filtro Kalman. 85

V.VIII Descripción funcional del sistema de seguimiento de objetivos móviles usando filtro Kalman. 95

V.IX Descripción de zonas del sistema de seguimiento de objetivos móviles usando filtro Kalman. 96

Conclusiones y trabajo futuro 110

Anexo A Código del sistema de seguimiento de objetivos móviles usando filtro Kalman. 112

Anexo B Cámara digital DFK 4303. 199

Anexo C Frame Grabber DFG-LC1. 209

Bibliografía 213

ÍNDICE DE FIGURAS

Figura 1	Estructura del ojo.	2
Figura 2	Clasificación de las imágenes.	4
Figura 3	Estructura de un sistema de visión por computadora.	6
Figura 4	Analogía ojo-cámara.	7
Figura 5	Ejemplo de visión por computadora.	9
Figura 6	Tarjetas capturadotas de imagen Frame Grabber.	11
Figura 7	Cámara CCD.	13
Figura 8	Forma de onda de video compuesto NTSC.	17
Figura 9	Forma de onda de video compuesto: Barras de color.	18
Figura 10	Comunicación serial síncrona.	21
Figura 11	Comunicación serial asíncrona.	23
Figura 12	Diagrama de pines del puerto paralelo.	24
Figura 13	Ejes coordenados de una imagen digital.	28
Figura 14	Etapas del procesamiento digital de imagen.	30
Figura 15	Ejemplo de niveles de muestreo y cuantificación en una imagen.	32
Figura 16	Esquema representativo de los niveles 4 y 8 de un píxel.	35
Figura 17	Diagramas de conectividad.	36
Figura 18	Histograma de una imagen.	38
Figura 19	Imagen segmentada.	40
Figura 20	Detección de bordes en una imagen.	42
Figura 21	Máscara general de dimensiones 3 X 3.	43
Figura 22	Umbralización utilizando la media aritmética del histograma.	44
Figura 23	Aplicación de un operador sobre una imagen.	48
Figura 24	Imagen procesada mediante un operador de tipo Sobel.	52
Figura 25	Predicción de la posición x_n y y_n.	55
Figura 26	Predicción del objeto filtrado y medición de la posición.	57
Figura 27	Filtro de Kalman.	63
Figura 28	Tarjeta digitalizadora de imagen o Frame Grabber.	77
Figura 29	Componente OCX de Imaging Source.	79

Figura 30	Etapas del proceso de análisis y obtención de resultados del sistema de seguimiento.	80
Figura 31	Diagrama de sistema de seguimiento.	81
Figura 32	Representación del formato estándar RGB 24 en un píxel.	82
Figura 33	Selección del objeto de interés.	83
Figura 34	Intensidad de niveles RGB representativo.	83
Figura 35	Obtención de la posición representativa.	84
Figura 36	Establecimiento de condiciones iniciales.	89
Figura 37	Predicción de la posición final del filtro de Kalman.	94
Figura 38	Pantalla principal de interfaz de usuario del sistema de seguimiento de objetos utilizando filtro Kalman.	95
Figura 39	Elemento archivo del menú e iconos relacionados.	97
Figura 40	Elemento configuración de imagen, pantalla de configuración de imagen e icono asociado.	98
Figura 41	Elemento de configuración de dispositivo, pantalla de configuración de dispositivo e icono asociado.	99
Figura 42	Elemento captura de video, icono asociado y pantalla de almacenamiento de video.	100
Figura 43	Icono de búsqueda de directorios y pantalla de exploración de ruta.	101
Figura 44	Elemento captura de imagen, icono asociado y pantalla de almacenamiento de imagen.	102
Figura 45	Elemento zoom e icono asociado.	103
Figura 46	Elemento histograma e icono asociado.	104
Figura 47	Elemento filtro e icono asociado.	105
Figura 48	Elemento escala de grises e icono asociado.	106
Figura 49	Elemento iniciar e icono asociado.	107
Figura 50	Elemento detener e icono asociado.	108
Figura 51	Elemento ayuda.	109

ÍNDICE DE TABLAS

Tabla (a)	Formatos de video analógico.	19
Tabla (b)	Pines del puerto paralelo.	25
Tabla (c)	Operadores gradiente más utilizados.	41

OBJETIVO DEL PROYECTO

Desarrollar un sistema computacional que con características similares a las del ojo humano, realice el seguimiento de un objeto mediante el modelo predictivo establecido por el *Filtro Kalman* en tiempo real.

HIPÓTESIS DEL PROYECTO

A partir de la captura mediante una cámara de video y utilizando el procesamiento y análisis de la imagen, será posible identificar un objeto dentro de una secuencia de imágenes logradas. Así mismo implementando el modelo predictivo de filtro Kalman se obtendrá el seguimiento de objetos en la imagen.

INTRODUCCIÓN

El ser humano se interrelaciona con el medio ambiente a través de sus diferentes sentidos, la vista, el tacto, el oído, el olfato y el gusto, que en conjunto conforman el mejor sistema experto. Resultan de tal importancia para la concepción del mundo que lo rodea, que de forma casi inconsciente son utilizados como una forma de lenguaje y como parte fundamental para la supervivencia.

De manera cotidiana y sin ningún esfuerzo, el ser humano abre los ojos para iniciar con ello el contacto con el exterior y es precisamente este sentido la base fundamental del presente trabajo de investigación.

Existen muchas razones por las cuales la naturaleza nos ha brindado la capacidad de ver o de tener un panorama real de las cosas que nos rodean. Con certeza, nuestra capacidad visual junto con nuestra capacidad de razonamiento nos han posibilitado para crear tecnología y ciencia. Resulta difícil imaginar el desarrollo científico y tecnológico sin las capacidades de visión que poseemos.

Uno de los desarrollos tecnológicos lo conforma el reconocimiento de objetos el cual es una tarea que el sistema de visión humana parece llevar a cabo sin ningún esfuerzo, sin embargo construir un sistema experto que realice esa misma función ha demostrado no ser tarea fácil.

El reconocimiento de objetos por computadora requiere implícitamente contar con los algoritmos de localización o seguimiento de objetos en escenas generalmente complejas y que cambian en el tiempo. La realización de esta función por medio de la computadora no es fácil, ya que existen variables que son difíciles de controlar en un sistema.

Existe un gran interés en la industria computacional por lograr equipos electrónicos inteligentes y de aprendizaje automático (*Machina Learning*), a lo cual los sistemas de visión inteligentes no son la excepción.

Los avances tecnológicos de los circuitos integrados (*hardware)* han provisto la capacidad para ser utilizados en el procesamiento de señales. Estos avances han permitido contar con la implementación de algoritmos para dicho

procesamiento en los circuitos integrados los cuales lo llevan a cabo en tiempo real a un costo relativamente bajo.

Por otro lado, se sabe que entre la electrónica y la computación existe una relación muy fuerte mantenida no sólo por el vínculo creado entre los componentes electrónicos, que son la base del funcionamiento de las computadoras, sino también por su funcionamiento interno y la programación, obligando a conocer las dos ramas para profundizar e investigar en temas comunes e íntimamente ligados, por lo que las personas dedicadas a la electrónica deben saber como utilizar los componentes destinados a la informática para poder hacer uso de ellos de manera óptima, es decir, poder utilizar los recursos computacionales a su favor. Lo mismo sucede con la gente de computación. Ésta debe saber los conceptos de electrónica y los diferentes usos de los dispositivos comunes existentes en la industria electrónica para poder auxiliarse de ellos en las aplicaciones o proyectos que se estén desarrollando.

Una de las labores más interesantes dentro del estudio de *Ingeniería Electrónica* e *Ingeniería en Computación* está representado por el desarrollo de interfaces con dispositivos y arreglos ya existentes para aplicaciones de interés específico o bien, el de crear nuevos instrumentos utilizando los diferentes desarrollos existentes.

En el presente trabajo de tesis se propone un modelo de seguimiento de objetos el cual comprende la integración de componentes ópticos, electrónicos, mecánicos y de procesamiento de imágenes.

Dicho modelo tendrá la capacidad de *recolectar* información a través de diversas técnicas de filtrado de imágenes teniendo como principal objetivo la implementación del *Filtro de Kalman*. El sistema desarrollado llevará a cabo observaciones diurnas de un objeto para su selección y seguimiento con condiciones especiales en tiempo *real*.

La estructura general de la actual tesis consiste en una introducción, cinco capítulos, una sección en la que se muestran las conclusiones de la misma y un apartado de anexos técnicos.

La introducción describe de forma genérica la problemática sobre la cual se desarrolla la tesis y la consecuente revisión de la estructura de la misma.

En el capítulo I se hace referencia a los conceptos sobre los que se basa el objetivo del presente trabajo de tesis.

En el capítulo II se presentan de forma genérica los conceptos del proceso de Visión utilizados en el campo de la computación.

En el capítulo III se presenta una introducción al procesamiento de imágenes.

En el capítulo IV se presenta la teoría de los diferentes filtros predictivos estableciendo la base técnica de la tesis.

En el capítulo V se muestra la implementación de los algoritmos de seguimiento utilizando programación en Delphi para la aplicación final.

Finalmente, se presenta una retroalimentación a manera de conclusiones las cuales engloban las funciones del proyecto y las futuras correcciones o adaptaciones que se podrían realizar al mismo para diversos usos y/o aplicaciones.

CAPÍTULO I

ANTECEDENTES

CAPÍTULO I

Las maravillas innumerables del universo nos son reveladas en la medida exacta con que somos capaces de percibirlas. La sutileza de nuestra visión depende de qué tan capaces somos de captarlas. La forma, la proporción y el orden son factores que determinan la belleza de todo aquello que a través de los ojos percibimos día tras día.

I.I Sistema de Visión Humana

El sentido de la vista es considerado con frecuencia por el ser humano como uno de los más valiosos. Para la mayoría de las personas la vista es su primer contacto con el exterior. Por su extraordinario diseño, los ojos nos permiten atestiguar la belleza del universo.

El sistema visual humano compuesto por el globo ocular (mejor conocido como ojo) y una porción del cerebro que procesa las señales neurológicas que provienen de éste forma un potente dispositivo de aprendizaje a partir de la recepción, almacenamiento y procesado de la información. Mediante procesos constantes de exploración y búsqueda e inserción de patrones visuales supone un sistema desarrollado de interacción con el medio.

Con más de doscientos cincuenta millones de receptores, el ojo humano posee una extraordinaria capacidad de resolución, puede apreciar más de dieciséis millones de colores y percibir matices de gran sutileza en la profundidad, color, forma y textura de los objetos, lo que le otorga una gran funcionalidad para el reconocimiento y la movilidad por el espacio y la supervivencia biológica.

El ojo de aproximadamente 2.5 cm. es un sensor capaz de detectar radiaciones de una longitud de onda de entre 400 y 700 nanómetros, en el rango conocido como luz visible, y de obtener datos instantáneos sobre la cercanía y lejanía de los objetos en función de diferentes parámetros como su tamaño, su colocación lineal, su gradiente textural, el paralaje, etc. Esto es posible en los seres humanos gracias a su disposición especial de los dos ojos a una distancia

aproximada de 6 cm., lo que permite no sólo un amplio campo de visión horizontal de aproximadamente 208° y unos 120° de visión vertical, sino también la posibilidad de visión tridimensional.

En la figura 1 se muestra la estructura física del ojo.

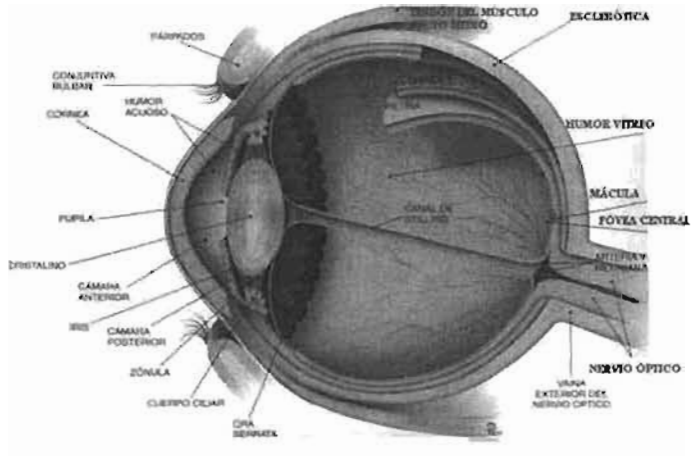


Figura 1 Estructura del Ojo humano

En la mayoria de las aplicaciones del procesamiento digital de imágenes, es necesario entender el sistema visual humano. El comprender las características y las limitaciones del sistema de visión puede ayudar a maximizar la efectividad de las operaciones en las aplicaciones realizadas.

Una tarea que el ser humano ha tratado de desarrollar en los últimos años es la de "plasmar" parte de su comportamiento en un objeto inerte como una máquina; un ejemplo de ello lo conforman los sistemas inteligentes, los cuales tratan entre otras cosas de tomar decisiones frente a situaciones diversas dadas ciertas condiciones. El sentido de la vista no escapa a ello.

Los sistemas expertos son sistemas informáticos que simulan el proceso de aprendizaje, de memorización, de razonamiento, de comunicación y de acción en consecuencia de un experto humano en cualquier rama de la ciencia. Estas características le permiten almacenar datos y conocimiento, sacar conclusiones lógicas, tomar decisiones, aprender de la experiencia y los datos

existentes, comunicarse con expertos humanos, explicar el por qué de las decisiones tomadas y realizar acciones como consecuencia de lo anterior.

Técnicamente, un sistema experto contiene una base de conocimientos que incluye la experiencia acumulada de expertos humanos y un conjunto de reglas para aplicar ésta base de conocimientos en una situación particular.

Los campos de desarrollo para los sistemas expertos son muchos y variados, se encuentran en el ámbito de la medicina, biología, economía, psicología, finanzas, procesamiento digital de señales y el procesamiento digital de imágenes, entre otros.

El elegir la vista como punto de partida para este trabajo de tesis tiene su razón de ser. El ser humano recibe la mayor parte de información de forma visual es decir; percibe por medio de los ojos, objetos mediante la acción de la luz descubriendo mediante imágenes las características de éstos.

A este proceso de recibir información se le denomina *percepción visual.*

Los *objetos* que son definidos mediante el proceso anterior son caracterizados mediante imágenes, las cuales no son sino una representación, semejanza o imitación de un objeto o cosa. Por tanto, y de forma general, una *imagen* constituye una representación del objeto: la imagen contiene información descriptiva que es utilizada para su representación.

Por lo anterior se clasifica a las imágenes dependiendo de la forma en la que son generadas.

- *Imágenes Físicas Visibles.* Perfectamente materiales y de naturaleza volátil o permanente. Las imágenes permanentes pueden ser ópticas constituidas por fotones en el dominio visible o imágenes electroópticas. Las imágenes permanentes son reproducciones de todo tipo: fotográficos, dibujos, pinturas, grabados, documentos impresos, etc.
- *Imágenes Físicas No Visibles.* Imágenes ópticas fuera del dominio visible o imágenes de naturaleza inmaterial: espectros físicos, mapas de poblaciones, representaciones de parámetros físicos no directamente visibles.

- *Imágenes Matemáticas.* Son preceptúales y por tanto invisibles por naturaleza. Pueden ser analógicas o digitales; representables mediante una función continua o una secuencia.

Haciendo uso de la teoría de conjuntos se puede representar la clasificación anterior en la figura 2.

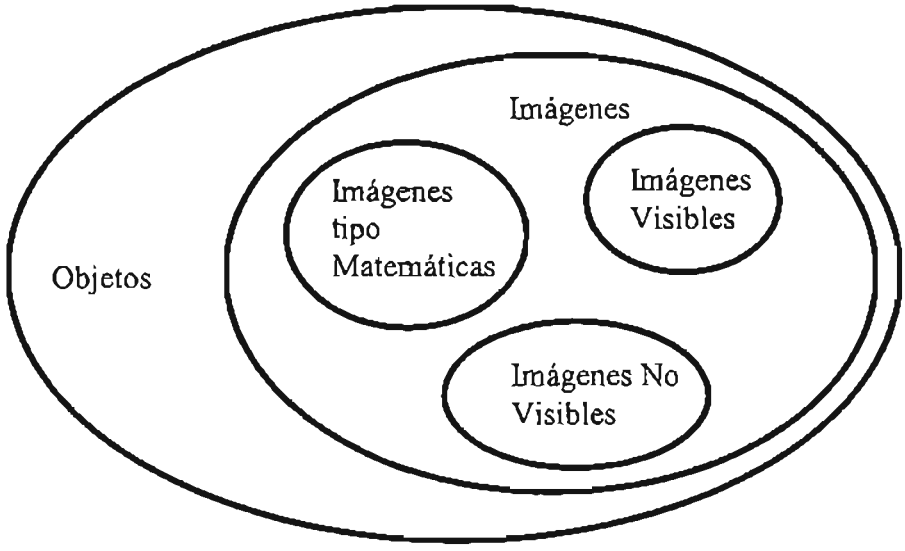


Figura 2 Clasificación de las Imágenes

I.II La Visión Humana y el Procesamiento Digital de Imágenes

La vista es el sentido más desarrollado que posee el ser humano por lo que no es de extrañar el desarrollo de nuevos sistemas informáticos capaces de reconocer, procesar e interpretar automáticamente todo tipo de gráficos, imágenes y videos. Las numerosas actividades del sector de la investigación han mostrado en los últimos años progresos decisivos, a pesar del altísimo poder computacional que requieren algunas de estas aplicaciones, como el procesamiento digital de imágenes, y el procesamiento digital de video.

Por lo anterior, la interrelación entre el ser humano y las máquinas ha tratado de asemejarse cada vez más a lo natural, sin embargo existen ciertas

limitantes, entre ellas se encuentran las limitaciones que presenta el ojo humano en lo que a la percepción se refiere. El ojo humano por un lado sólo puede captar un cierto tipo de energía dentro de lo que se llama *rango visible*, el calor y las microondas, que son otras formas de energías no pueden ser percibidas directamente y por otro lado, un ser humano no puede ver más allá de lo que su estatura le permite.

De cualquier forma el ser humano posee una visión oblicua por lo que no puede captar objetos a grandes distancias por lo que requiere de cierto tipo de ayuda.

Gran parte de esa ayuda la forman herramientas que entre otros hacen uso del llamado *procesamiento digital de imágenes*.

El procesamiento digital de imágenes es una disciplina que desarrolla las bases teóricas y algorítmicas mediante las cuales puede extraerse información del mundo real de forma automática a partir de una imagen observada. El espectro de aplicaciones es amplio e incluye desde aplicaciones industriales, imágenes aéreas e imágenes médicas entre otros.

Una importante aplicación de la teoría del procesamiento de imágenes está directamente relacionada con la visión humana, como resultado el procesamiento de imágenes tiene un gran número de aplicaciones desempeñando un gran papel en la vida cotidiana.

El procesamiento digital de imágenes se puede clasificar básicamente en cuatro grandes áreas:

- Realce de imágenes. Proceso mediante el cual se acentúan ciertas partes de la imagen para un posterior análisis. Ejemplos típicos son el contraste, detección de bordes, filtrado de ruido, etc.
- Restauración de imágenes. Elimina o minimiza la degradación de las imágenes. Lo anterior incluye la borrosidad generada por un sensor o ambiente, filtrado de ruido y corrección de la distorsión geométrica.

- Codificación de imágenes. Puede ser usada para reducir el ancho de banda de un canal de comunicación cuando una imagen es transmitida.
- Análisis de imágenes. Representa simbólicamente el contenido de la imagen. Las aplicaciones incluyen visión por ordenador, robótica e identificación de objetos. En el análisis de imágenes la entrada es una imagen pero la salida es típicamente una representación simbólica del contenido de la imagen de entrada.

El utilizar procesamiento digital de imágenes a lo largo de este trabajo de tesis permitirá manipular las imágenes captadas a través de una cámara logrando con ello resaltar las características más importantes para el cumplimiento del objetivo.

De esta forma, en la figura 3 se muestra la estructura general del sistema de visión por computadora que buscará realizarse.

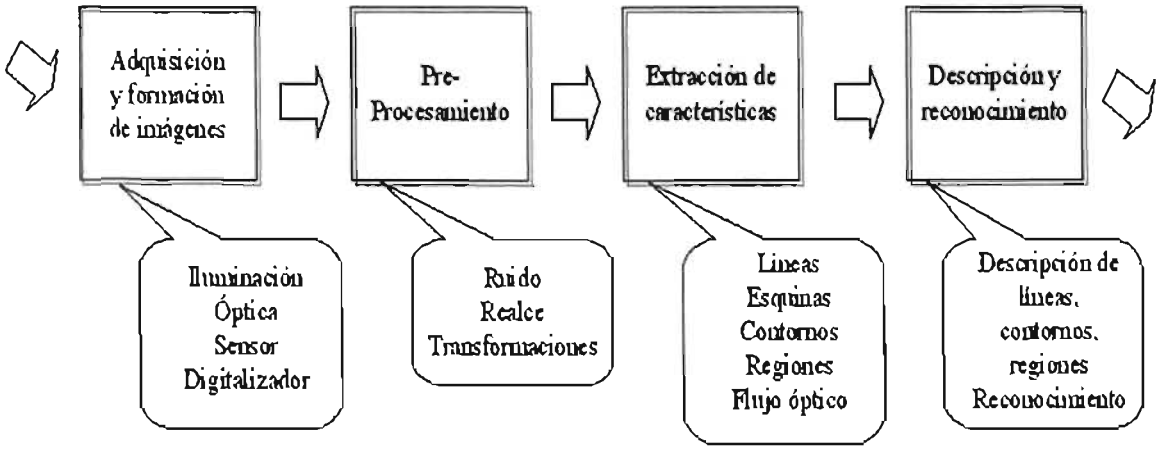


Figura 3 Estructura de un sistema de visión por computadora

I.III Analogía ojo-cámara

Aunque el funcionamiento del ojo humano ha tratado de ser replicado en el funcionamiento de una cámara, aún existen ciertas limitantes. La figura 4 muestra las principales diferencias de acuerdo a los elementos señalados. Además de ellas, es vital señalar que una de las ventajas del sistema visual humano es que con los dos ojos se hace posible la visión binocular que ayuda a percibir las relaciones espaciales entre los objetos, dando como resultado la tercera dimensión. La imagen obtenida por una cámara sólo es bidimensional.

El sistema de visión humano permite la percepción visual del movimiento, aspecto que no es captado a la misma velocidad en una cámara.

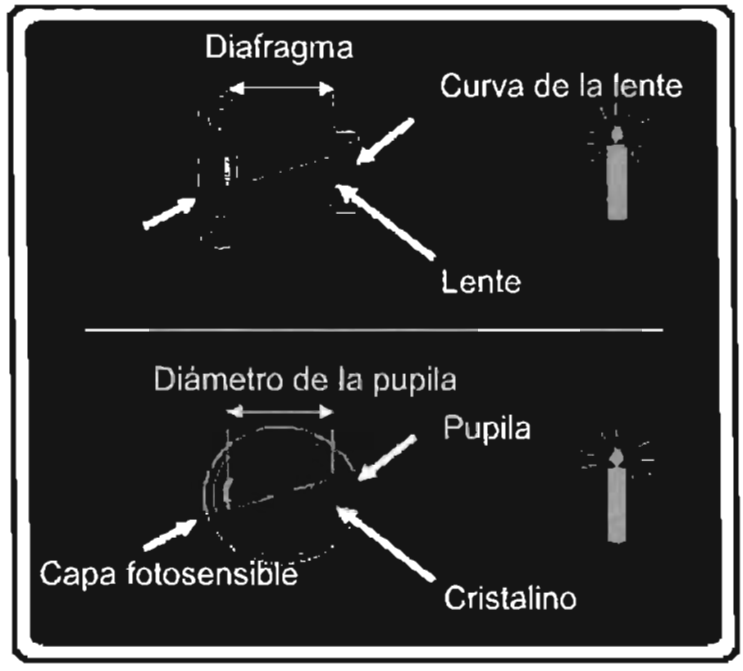


Figura 4 Analogía Ojo-Cámara

Algunas de las diferencias existentes entre el funcionamiento del ojo y el de una cámara se explican a continuación a manera de ejemplo.

El *cristalino*, que se mueve equivalente a la *lente* de la cámara, se encuentra en una posición fija no se mueve hacia delante y hacia atrás como en la cámara. Para poder enfocar, el ojo humano utiliza los músculos que halan y hacen que cambie la curvatura.

El diámetro de la *pupila*, equivalente al *diafragma* de la cámara, permite funcionar en un rango mucho más amplio de intensidades de iluminación que la cámara.

La *capa fotosensible* en el ojo humano es curva, lo que compensa la curvatura de la lente y por tanto permite un enfoque más rápido y eficaz.

La *distribución de los conos y bastones* hacen que el grado de sensibilidad en la retina no sea homogéneo, no así en la cámara.

I.IV Requerimientos de hardware y software para los sistemas de visión por computadora

El término *visión por computadora* dentro del campo de la inteligencia artificial es considerado como el conjunto de todas aquellas técnicas y modelos que permiten procesar, analizar y explicar cualquier tipo de información obtenida a través de imágenes digitales. Desde sus inicios la visión por computadora ha inspirado sus desarrollos en el estudio del sistema visual humano el cual sugiere, la existencia de diferentes tipos de tratamientos de la información visual dependiendo de objetivos específicos es decir, la información visual percibida es procesada en distintas formas con base en las características particulares de la tarea a realizar, por lo que la visión por computadora propone varias técnicas que permiten obtener una representación del mundo a partir del análisis de imágenes obtenidas desde cámaras de video.

Debido a que la información visual es una de las principales fuentes del mundo real, resulta útil proveer a una computadora digital del sentido de la vista (a partir de imágenes tomadas con cámaras digitales o analógicas), que junto con otros mecanismos como el aprendizaje hagan de ésta una herramienta

capaz de detectar y ubicar objetos en el mundo real, objetivo principal de la visión por computadora.

La figura 5 ejemplifica el proceso de visión por computadora.

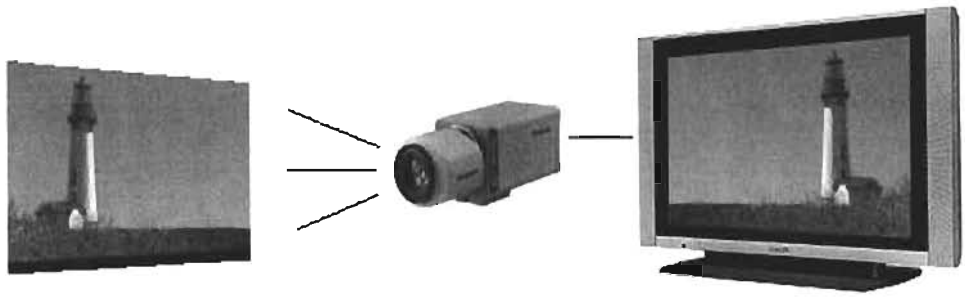


Figura 5 Ejemplo de visión por computadora

La aplicación de la visión por computadora en la realización de trabajos de inspección tiene como ventaja la utilización de un sensor remoto (comúnmente ubicado en cámaras de video) que no requiere ningún contacto físico con los objetos a visualizar.

Como se mencionó anteriormente, la visión por computadora requiere de componentes de hardware tales como cámaras digitales o analógicas y en ocasiones, componentes adicionales que permitan una mejor recopilación de la información tales como los llamados *Frame Grabber* o digitalizadores de imagen.

I.V Cámara de video analógica

Las etapas de operación de una cámara suelen dividirse en etapa de adquisición de imagen y etapa de formato de salida de video.

Las cámaras actuales basan su funcionamiento en chips CCD (Charge Coupled Device) que son matrices bidimensionales de elementos fotosensibles

similares a capacitores que almacenan carga eléctrica en función de la intensidad luminosa incidente.

Las dimensiones de dicha matriz de sensores (columnas X filas) definen la resolución de la cámara.

El orden y frecuencia en que la matriz de sensores CCD es *barrida* y la forma en que es traducida y codificada la carga de los sensores del CCD, definen la segunda etapa de operación de la cámara, que se encarga de dar formato a la salida de video.

I.VI Tarjeta capturadora de imagen

Las tarjetas capturadoras de imagen, son dispositivos periféricos utilizados por la computadora para mostrar imágenes en pantalla, capturando la señal en formato analógico y transformándolo a formatos digitales tales como AVI, MPEG, etc. Estas tarjetas suelen diferenciarse por la cantidad de puntos de colores que muestran, la rapidez y la capacidad de memoria de que disponen (a mayor cantidad de memoria, será capaz de mostrar imágenes con más puntos y/o colores).

El formato más común y soportado por todo tipo de capturadoras de video es el RGB de 24 bits. Las tarjetas capturadoras de imagen con formato RGB son capaces de obtener casi el 100% de la calidad de una imagen.

Al referir el formato RGB se suele especificar también la profundidad de color, que se expresa en bits. Así, por ejemplo, al decir RGB 24 se indica que el video tiene una profundidad de color de 24 bits (16,777.216 colores). Los valores posible son RGB 32, RGB 24, RGB 16 y RGB 15.

Para el desarrollo del presente trabajo de tesis, se utilizó una tarjeta capturadora de imagen mejor conocida como *Frame Grabber (ver Capítulo V)*.

La tarjeta capturadora de imagen Frame Grabber desarrollada a inicios de los años 80's permite la captura de una imagen estática (frame) procedente de una fuente de video.

Su conexión era un poco peculiar en sus inicios. Primeramente se conectaba al *Amiga* a través del puerto paralelo, luego se le conectaba la fuente de video a digitalizar y por último a un monitor para ver dicha digitalización.

Podía capturar imágenes en escala de grises (256 colores) o a 24 bits de color. Esto último presentaba un inconveniente, para realizar la captura en color, el video debía tener una salida RGB. Posteriormente, salió una versión que incorporaba un separador[1] de colores para permitir la captura en color.

La figura 6 presenta dos tarjetas capturadoras de imagen, la primera una tarjeta comercial y la segunda la tarjeta *frame grabber*.

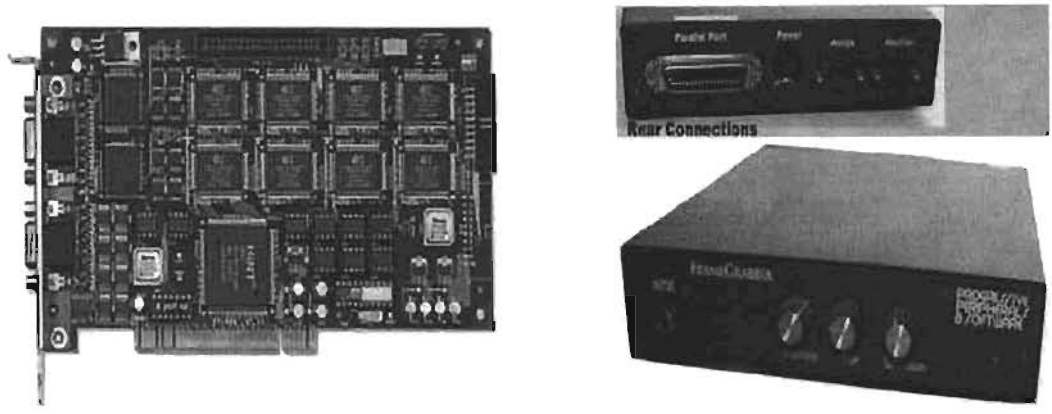


Figura 6 Tarjetas capturadoras de imagen Frame Grabber

[1]Separador de colores conocido como *Splitter*.

I.VII Cámaras CCD

Las cámaras que se usan en el campo del procesamiento digital de imágenes constan primordialmente de dos partes:

a) Unidad de Adquisición de Imagen.

b) Unida de Salida de Imagen.

La unidad de adquisición queda definida por el tipo de sensor que registra la señal luminosa que conforma la imagen.

La unidad de salida es la encargada de interpretar los valores del sensor y convertirlos en una señal eléctrica de video conforme a un protocolo el cual puede ser estándar o no, dicho protocolo define entre otros parámetros el número de líneas horizontales y verticales por cuadro así como el número de cuadros por segundo.

Los protocolos estandarizados son por ejemplo, los reconocidos por organismos internacionales como el CCIR[2] para Europa y EIA[3] para América y que se ajustan a monitores analógicos. En tanto que los protocolos no estandarizados tienen una orientación hacia cámaras digitales.

- Adquisición de Imagen

Las cámaras CCD (Charge Coupled Device) basan su funcionamiento en un chip que consiste de un arreglo bidimensional[4] de sensores de señales luminosas. La dimensión de dicho arreglo define la resolución de la imagen que se obtenga a la salida de la cámara.

[2]CCIR Comité Consultatif Internacional Des Radiocommunications (Comité Consultivo Internacional de Radiocomunicaciones).

[3]EIA Electronics Industruies Association (Asociación de Industrias Electrónicas).

[4]Existen también las cámaras de escaneo progresivo que cuentan sólo con una hilera de sensores que realizan un barrido para obtener una imagen bidimensional.

Cada sensor del chip CCD transforma la luz (longitudes de onda entre 400nm y 1000nm para el espectro visible) en una carga, la cual a su vez es transformada en un voltaje. Los fotones incidentes en el sensor generan electrones que se acumulan haciendo que éste funcione como un capacitor entregando un voltaje acorde a la cantidad de carga acumulada.

La figura 7 muestra un tipo de cámara CCD.

Figura 7 Cámara CCD

Además de la luz incidente, la temperatura ambiente genera electrones espurios, la llamada *corriente oscura*, lo cual origina efectos no deseados conocidos como *ruido*.

Como resultado, un chip CCD se puede *llenar* de electrones aún sin ser expuesto a alguna fuente de luz. Este efecto no deseado se contrarresta adicionando sistemas de enfriamiento a los chips, o en el caso de tomas estáticas, obteniendo varios cuadros y promediándolos para reducir el ruido (considerando que el ruido sea un evento aleatorio de media cero)

- Parámetros de importancia de la cámara CCD

El tiempo de exposición del chip CCD es el parámetro que determina la cantidad de tiempo que los sensores están expuestos a la señal luminosa.

El método tradicional de controlarlo para cámaras analógicas convencionales es con el uso de un obturador mecánico, pero en el caso de las

cámaras CCD, el dispositivo que controla la exposición es un *obturador electrónico* también conocido como *integrador.*

Así mismo, para la tecnología CCD en lugar de utilizarse el término tiempo de exposición éste es sustituido por *tiempo de integración*, el cual es más descriptivo tomando en cuenta la naturaleza capacitiva de los sensores CCD.

Si la intensidad de la luz recibida es muy baja o muy alta, el resultado a su vez serán imágenes pobres o saturadas. La calidad de la imagen se puede manipular variando la ganancia que es parte de la generación de la señal, o modificando el tiempo de integración.

Otro de los parámetros importantes que deben de ser considerados al trabajar con el chip CCD es su *resolución* expresada en el número de píxeles.

Una resolución típica en este chip es por ejemplo 756 X 581 (columnas por líneas). El número total de píxeles suele ser mayor, lo cual se explica debido a que el chip tiene algunas líneas y columnas de píxeles ciegos conocidos como área de oscuridad óptica

La *respuesta espectral* es otro parámetro del chip CCD el cual indica el rango de frecuencias (o longitudes de onda) a las cuales el chip es sensible.

Los chips CCD no tienen la capacidad inherente de discriminar el color (esto es, diferenciar las distintas longitudes de onda dentro de su espectro de respuesta). Por lo tanto, en las cámaras a color la luz debe separarse en sus colores primarios verde, rojo y azul (esquema RGB) por medio de filtros y prismas. La separación de los componentes de color para cámaras CCD sigue dos esquemas principales.

El primero es separar los componentes RGB[5] por medio de un prisma que dirige cada componente de color a un chip CCD separado. Las cámaras con está tecnología se llaman cámaras 3CCD, representando una mayor calidad pero también un incremento es su costo.

[5] RGB. Modo de representación de colores, proporciona el mayor espectro visible mezclando las luces generadas por fósforos rojos, verdes y azules.

Otra aproximación al color más simple y de un menor costo utiliza una estructura de pequeños filtros llamados *filtros de mosaico*, esta aproximación distribuye cada componente de color de manera alternada en los píxeles de un solo chip, esto es; se tiene tres píxeles vecinos[6] cada uno con los filtros necesarios que le hacen responder sólo a la componente de luz roja, verde y azul alternadamente.

I.VIII Unidad de salida de una cámara de video

La unidad de salida de una cámara genera la señal eléctrica apropiada para que el dispositivo subsiguiente (tarjeta de video, monitor analógico, monitor digital, Frame Grabber, etc.) realice con ella las operaciones adecuadas y requeridas.

Dado que primordialmente la señal de salida de una cámara va aun monitor analógico, ésta suele apegarse a alguno de los estándares internacionales tales como: PAL (Phase Alternation Line) y SECAM (Séquentiel Couleur á Mémoire) para CCIR, RS-170 y NTCS para EIA.

La unidad de salida se encarga de interpretar el nivel de carga en cada uno de los *píxeles* del chip CCD y asignarles una señal la cual contiene información de tono e intensidad. Dicha señal es "montada" sobre otra señal portadora junto con pulsos de sincronía vertical y horizontal que indican el comienzo de un nuevo cuadro (o medio cuadro)[7] y el comienzo de una nueva línea respectivamente.

El estándar de video determinará el numero de señales necesarias para codificar toda la información de video, el número de píxeles por línea, las líneas por cuadro y los cuadros por segundo.

[6]El concepto "vecinos de un píxel" será abarcado en el capítulo II.

[7]Depende de si la señal es trenzada o no.

I.IX Escaneo progresivo y trenzado

Para formar una imagen en un monitor CRT (Catodic Ray Tube) ya sea digital o analógico, un grupo de cañones de electrones (tres para el monitor a color, uno para cada color primario) controlados por la señal de video, proyectan un haz de electrones contra una pantalla luminiscente situada en el extremo opuesto de los tubos.

Los haces de electrones "barren" la pantalla formando así la imagen.

Existen dos tipos de sistemas de barrido de imagen que difieren en la técnica que utilizan para pintar una imagen completa en una pantalla. Los monitores compatibles con señales de televisión generalmente utilizan la técnica de *trenzado*, en tanto que los monitores digitales de las computadoras aplican la técnica de *barrido progresivo*. Este par de formatos son incompatibles lo cual hace necesario que uno de ellos sea convertido al formado del otro para poder hacer operaciones comunes.

El *barrido trenzado* consiste en que cada imagen referida como cuadro sea dividida en dos subimágenes referidas como campos. Un campo está formado por las líneas horizontales "pares" de la imagen y el otro a las líneas horizontales "nones". La señal de video controla los haces de electrones para que impresionen todo un campo non en la pantalla, seguido de un campo par, esto es; se forma una imagen completa en dos barridos.

El *barrido progresivo* consiste en formar la imagen de forma corrida, es decir se barren todas las líneas horizontales en una sola pasada.

I.X Forma de onda típica de video analógico

En términos generales, una señal de video analógica estándar es de tipo compuesto, lo que significa que en una sola señal eléctrica se "montan" todos los parámetros que describen a las imágenes. Dicha señal combina la información de brillantez (*luma*), la información de color (*chroma*), e información de sincronización en una sola señal.

La figura 8 muestra una señal compuesta NTCS que representa una línea horizontal de color blanco.

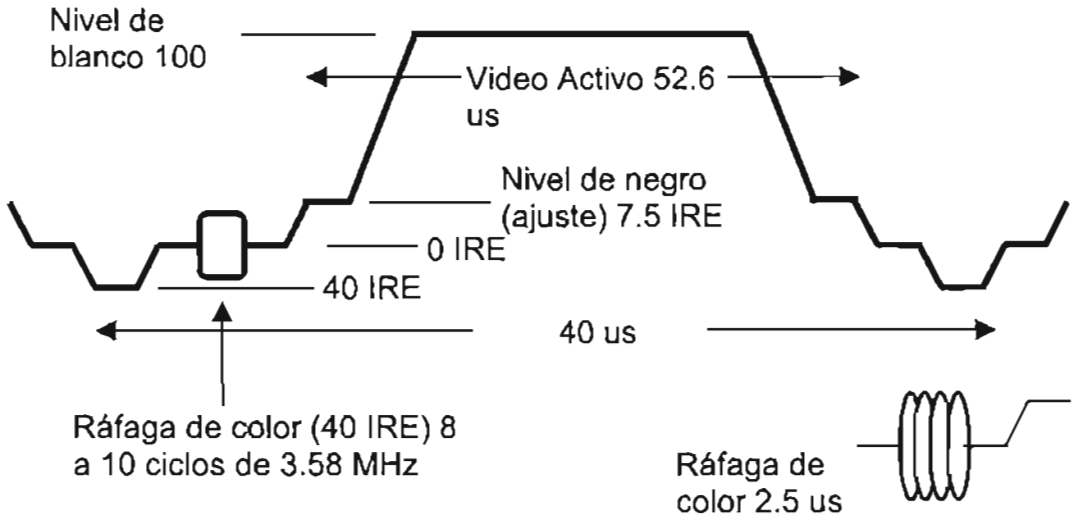


Figura 8 Forma de onda de video compuesto NTSC

Cada línea esta conformada por una porción activa que indica la intensidad y color de cada píxel de la línea y una porción inactiva o de recuperación horizontal en la cual, el haz de electrones inactivo se posiciona al inicio de la siguiente línea y se envían las referencias de brillo y color.

La brillantez está dada por la amplitud instantánea de la porción de video activo. La unidad de medida de la amplitud se da en términos de IRE's.

IRE es una medida arbitraria en la que por ejemplo, 140 IRE= 1 V. En la figura 7 se establece el nivel de referencia del negro a 7.5 IRE.

La información de color se agrega sobre el nivel de brillantez y es una onda senoidal cuya diferencia de fase respecto a la fase de la ráfaga de color (enviada antes de la porción activada) determina el color.

Lo anterior se representa en la figura 9.

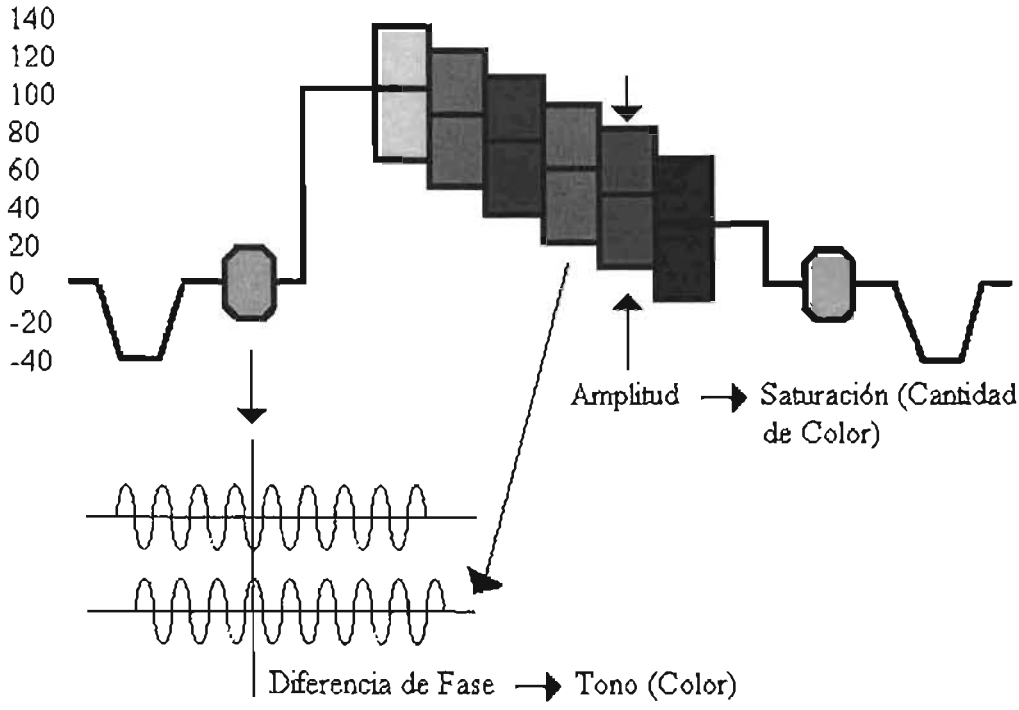


Figura 9 Forma de onda de video compuesto: Barras de color

I.XI Tabla de Formatos de video analógico

El objetivo de la mayoría de los sistemas de video analógico es la distribución de programas e información, para ello se requiere que los equipos estés estandarizados de forma que las señales generadas puedan ser utilizadas por cualquier persona.

La tabla (a) muestra los principales formatos de video analógico

Formato de video	NTSC	PAL	HDTV/SDTV	VGA	XGA
Descripción	Formato de televisión para Norteamérica y Japón	Formato de televisión para la mayoría de Europa y Sudamérica	Alta definición (HD) / definición estándar (SD) para televisión digital	Video graphics Array (PC)	Extended Graphics Array (PC)
Resolución vertical (líneas visibles por cuadro)	Aprox. 480 (525 líneas totales)	Aprox. 575 (625 líneas totales)	1080 ó 720 ó 480; 18 formatos distintos	480	768
Resolución horizontal (píxeles visibles por línea)	Determinado por el ancho de banda, va de 320 a 650	Determinado por el ancho de banda, va de 320 a 720	1920 ó 704 ó 640; 18 formatos distintos	640	1024
Frecuencia horizontal (kHZ)	15.734	15.625	33.75-45	31.5	60
Cuadros por segundo	29.97	25	30-60	60-80	60-80
Ancho de banda	4.2	5.5	25	15.3	40.7

(a) Formatos de video analógico

I.XII Comunicación entre dispositivos digitales

Los puertos de E/S constituyen el medio por el cual el microprocesador de una computadora se comunica con su entorno. Existen puertos para cada interacción de la unidad de procesamiento principal con sus dispositivos auxiliares.

I.XIII Comunicación serial

La comunicación serial, establece entre dos dispositivos un intercambio de datos bit por bit. Esto significa que cada palabra, cada byte de información, se transmite descomponiéndolo en sus bits constitutivos y enviando sucesivamente cada uno de ellos.

La comunicación serial se presenta en dos formas básicas para los microcontroladores y microprocesadores: síncrona y asíncrona.

En la comunicación síncrona, los dispositivos comunicados se sincronizan a nivel de palabra esto es, se establece el instante preciso en que las palabras deben ser transmitidas o recibidas. Por otro lado, en la comunicación serial asíncrona, la recepción o transmisión de cada palabra es aleatoria; lo que se sincroniza es el instante preciso en el que cada bit es transmitido o recibido.

Para establecer cualquiera de los dos tipos de comunicación serial, es necesario adoptar un código que permita la transferencia de datos. Hablando de bits, se tienen dos parámetros que pueden definir su estado: 0 ó 1, alto o bajo. Cada palabra de un código consta de *n* bits, de donde el número de palabras del código es igual a $2^n - 1$. A este número de bits se le puede agregar bits de paridad para detección o corrección de errores.

I.XIV Comunicación serial síncrona

La comunicación serial síncrona es aquella en la que los dispositivos de envío y recepción de la comunicación son sincronizados utilizando un reloj que cronometra con precisión el tiempo que separa cada bit.

Dado que los bits se envían y reciben en un proceso controlado (sincronizado) no se requiere de los bits de inicio y final como en el caso de la comunicación asíncrona.

Las transmisiones se detienen cuando se alcanza el final de una trama lo que refiere una mayor eficiencia que la transmisión asíncrona, especialmente cuando se están transfiriendo grandes paquetes de datos.

En el caso de que se genere un error durante la transmisión de datos, el esquema de corrección y detección genera automáticamente una retransmisión de los mismos.

La comunicación síncrona se utiliza en la mayoría de las comunicación de red y digitales.

La figura 10 presenta un esquema general de la comunicación síncrona.

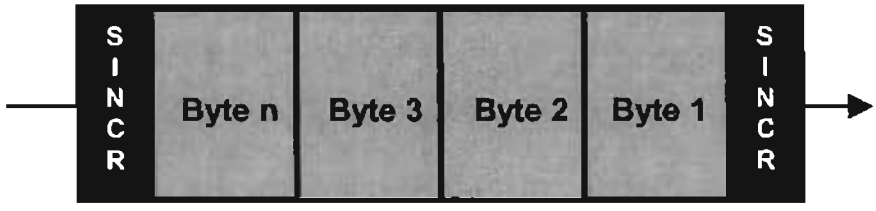


Figura 10 Comunicación serial síncrona

I.XV Comunicación serial asíncrona

La comunicación serial asíncrona comúnmente se resuelve con un dispositivo UART (Universal Asynchronous Receiver/Transmitter) que recibe en forma paralela un byte de información a transmitir y lo envía en forma serial utilizando alguno de los protocolos de transferencia serial como el RS-232, usado en forma genérica por los puertos seriales de las computadoras.

Este protocolo maneja dos tipos de códigos llamados *marca* y *espacio.* El estado alto de voltaje se considera como el uno lógico, y el estado bajo es considerado como 0 lógico. La diferencia entre los dos códigos radica en cual de los estados es el que se mantiene estable, esto es; cual de los estados es el que se tiene cuando no hay transmisión de datos.

A la comunicación serial asíncrona la definen los siguientes parámetros:

- Velocidad de transmisión (bits por segundo).
- Longitud de palabra.
- Bits por paro
- Bits de paridad.

De esta manera al recibir el bit de inicio se sabe cuantos bits subsecuentes constituyen la palabra de datos y cuantos bits son sólo de control.

Se transmiten siempre el mismo número de bits, ello es; un número de bits idéntico para cada palabra transmitida. El inicio de la transmisión de los datos es totalmente asíncrona, pues no se sabe el momento exacto en el que comienza la transmisión, pero una vez que haya comenzado se sabe cuanto durará debido a que el primer bit que se envía siempre es un bit de inicio que hace que cambie el estado *estable* de la comunicación, después se envía cierto número de bits que es la palabra que contiene la información y posteriormente se pueden incluir bits de paridad para seguridad, se termina la transmisión con un bit de paro[8] que es del mismo valor lógico que el bit de arranque.

[8]El parámetro bit de paro toma comúnmente los valores 1,1.5 y 2.

Los datos son transmitidos del bit menos significativos (LSB) al bit más significativo (MSB). Obviamente el bit menos significativo es precedido por el bit de arranque. Los bits de paridad o detección de errores son agregados después del MSB y el último en transmitir es el bit de paro indicando la separación entre una palabra y la siguiente.

La figura 11 representa el esquema de comunicación serial asíncrona.

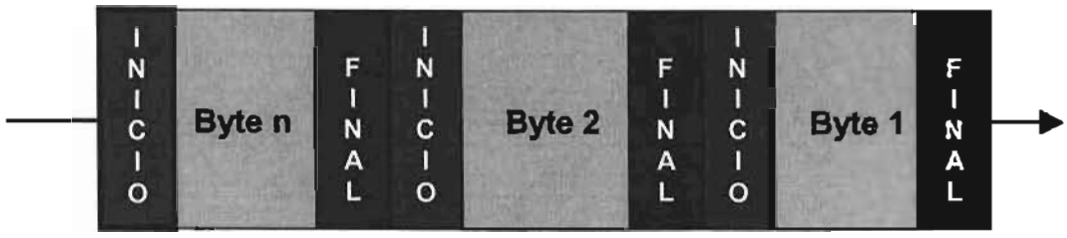


Figura 11 Comunicación serial asíncrona

I.XVI Comunicación Paralela

El método de transferencia paralela entre dispositivos digitales se basa en múltiples líneas físicas de datos, esto permite enviar y recibir simultáneamente un conjunto de varios bits.

El puerto paralelo se utiliza generalmente para manejar impresoras. Sin embargo, dado que tiene un conjunto de entradas y salidas digitales se puede emplear para leer datos y controlar dispositivos en general. De esta forma, el puerto paralelo se puede utilizar como interfaz de entrada y salida que funcione subordinado a rutinas de software.

El puerto paralelo se identifica por su dirección de I/O (entrada/salida) base y se identifica ante sistemas MSDOS por el número LPT. Las direcciones estándar para el puerto son 03BCh, 0378h y 0278h.

El puerto consiste en tres registros de 8 bits ubicados en direcciones adyacentes del espacio de I/O de la PC. Los registros se definen relativos a la dirección de I/O base y son:

- IOBase+0:Registros de datos.
- IOBase+1:Registros de estado.
- IOBase+2:Registros de control.

El *registro de datos* permite escribir (o leer si el puerto es bidireccional) un dato en los pines 2 al 9 del conector del puerto. SI se lee el registro, se verifica el último dato escrito (o el último dato recibido si el puerto es bidireccional). La corriente que pueden entregar los pines es de 2.6 mA. y pueden drenar un máximo de 24 mA.

El *registro de estado* conformado por cinco bits, es únicamente de lectura.

El *registro de control* conformado por 4 bits, es únicamente de escritura.

La figura 12 muestra la estructura del puerto paralelo.

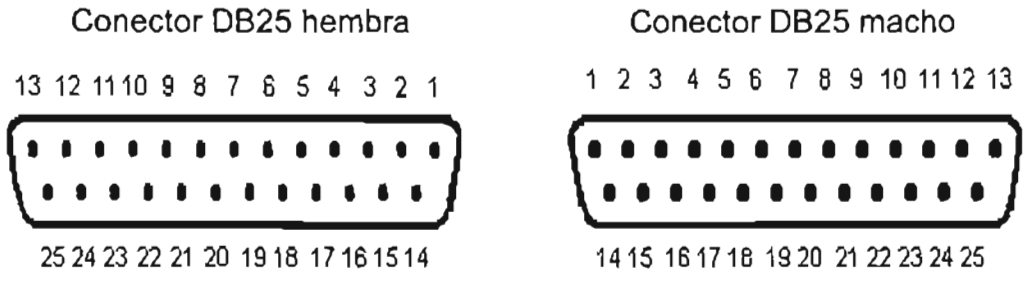


Figura 12 Diagrama del puerto paralelo

La tabla (b) describe cada uno de los pines que componen al puerto paralelo.

Descripción de los Pines del Puerto Paralelo					
DB25	**Señal**	**Registro**	**Tipo**	**Activo**	**Sentido**
1	Control 0	C0-	Salida	Bajo	Invertido
2	Dato 0	D0	Salida	Alto	directo
3	Dato 1	D1	Salida	Alto	directo
4	Dato 2	D2	Salida	Alto	directo
5	Dato 3	D3	Salida	Alto	directo
6	Dato 4	D4	Salida	Alto	directo
7	Dato 5	D5	Salida	Alto	directo
8	Dato 6	D6	Salida	Alto	directo
9	Dato 7	D7	Salida	Alto	directo
10	Estado 6	S6+	Entrada	Alto	directo
11	Estado 7	S7-	Entrada	Bajo	Invertido
12	Estado 5	S5+	Entrada	Alto	directo
13	Estado 4	S4+	Entrada	Alto	directo
14	Control 1	C1-	Salida	Bajo	Invertido
15	Estado 3	S3+	Entrada	Alto	directo
16	Control 2	C2+	Salida	Alto	directo
17	Control 3	C3-	Salida	Bajo	Invertido
18-25	Tierra				

(b) Pines del puerto paralelo

Las direcciones I/O de los registros del puerto paralelo se almacenan en una tabla de manera que se pueda obtener acceso a ellos por medio de distintos lenguajes tales como Pascal, Quick Basic, Windows, C, Delphi, etc.

I.XVII Comunicación USB

USB o Universal Serial Bus es una interfaz para transmisión de datos y distribución de energía creada principalmente con el fin de mejorar la velocidad de transmisión de datos en las interfaces serie y paralelo. Es un interfaz de 4 hilos, 12 Mbps y "plug and play" que distribuye 5V para alimentación de los dispositivos electrónicos.

El Universal Serial Bus como su nombre lo indica, es un bus serie que hace posible la conexión de hasta 127 periféricos a una única PC, con detección y configuración automáticas.

Es un bus basado en el paso de un testigo. El controlador USB distribuye testigos por el bus de forma que el dispositivo cuya dirección coincide con la que porta el testigo responde aceptando o enviando datos al controlador el cual también gestiona la distribución de energía a los periféricos que lo requieren.

A diferencia de otras arquitecturas, USB no es un bus de almacenamiento y envío por lo que no existe retardo en el envío de un paquete de datos.

CAPÍTULO II

CONCEPTOS GENERALES DEL PROCESAMIENTO DE IMÁGENES

CAPÍTULO II

El interés por los métodos de tratamiento digital de imágenes deriva de dos áreas principales de aplicación: la mejora de la información pictórica para la interpretación humana y el procesamiento de los datos de la escena para la percepción automática por una máquina.

El procesamiento de imágenes tiene como objetivo mejorar el aspecto de las imágenes y hacer más evidentes en ellas algunos detalles.

La imagen pudo haber sido generada de muchas formas por ejemplo, fotográficamente o, electrónicamente por medio de monitores de televisión. El procesamiento de las imágenes se puede en general hacer por medio de métodos ópticos, o bien por medio de métodos digitales tales como una computadora.

II.I Representación digital de imágenes

Una imagen es una representación de dos dimensiones del mundo visual. El término imagen monocroma o simplemente imagen se refiere a una función bidimensional de intensidad de la luz $f(x,y)$, donde x y y representan las coordenadas espaciales y el valor de f en un punto cualquiera (x,y) es proporcional al brillo (o nivel de gris) de la imagen en ese punto.

Una imagen digital es una imagen $f(x,y)$ que se ha discretizado tanto en las coordenadas espaciales como en el brillo. Una imagen digital puede considerarse como una matriz cuyos índices de fila y columna identifican un punto de la imagen y el valor del correspondiente elemento de la matriz indica el nivel de gris en ese punto. Los elementos de una distribución digital de este tipo se denominan elementos de la imagen, pixeles o pels.

La figura 13 muestra un ejemplo de las coordenadas de una imagen.

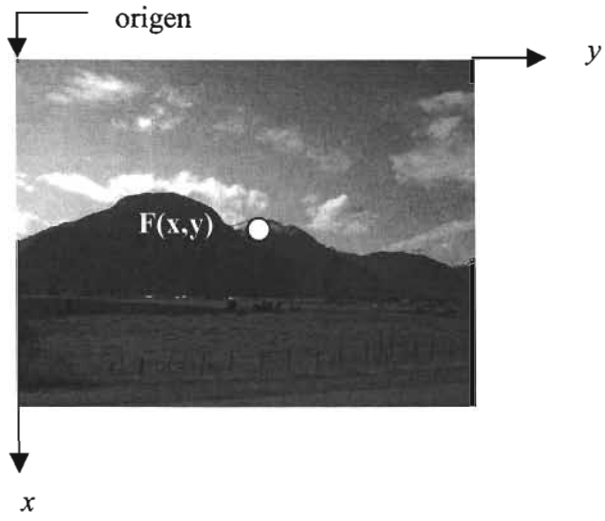


Figura 13 Ejes de una imagen digital

II.II Fases del procesamiento digital de imágenes

El tratamiento digital de imágenes comprende un amplio rango de hardware, software y recursos teóricos. Es importante aclarar que aunque no es parte del procesamiento digital de imágenes el tener una base de conocimientos y/o un dominio del problema es parte importante y fundamental para desarrollar las etapas que lo componen, el procesamiento digital de imágenes está integrado por cinco etapas básicas.

La primera etapa del proceso es la adquisición de la imagen es decir, la adquisición de una imagen digital. Para ello se necesita un sensor de imágenes y la posibilidad de digitalizar la señal producida por el sensor, esta parte la obtendremos por medio de una cámara de vigilancia.

La segunda etapa trata del preprocesamiento de la imagen previamente capturada. La función básica del preprocesamiento es la de mejorar la imagen, dicho de otra forma, el procesamiento trata típicamente las técnicas de mejorar el contraste, eliminar el ruido y aislar las regiones cuya textura indica la probabilidad de información alfanumérica.

La tercera etapa es la segmentación. Consiste en partir la imagen de entrada en sus partes constituyentes u objetos. A la salida del proceso de segmentación se tienen los datos de un píxel en bruto, que constituyen bien el contorno de una región o bien todos los puntos de una región determinada.

La cuarta etapa es la representación y descripción. La representación consiste en elegir la forma en la que se presentaran los datos obtenidos: como contorno o como región completa. La descripción también denominada selección de rasgos consiste en extraer rasgos de una información cuantitativa de interés o que sean fundamentales para diferenciar una clase de objetos de otra.

La quinta y última etapa conste en el reconocimiento y la interpretación.

El reconocimiento es el proceso que asigna una etiqueta a un objeto basándose en la información proporcionada por sus descriptores. La interpretación implica asignar significado a un conjunto de objetos reconocidos.

La utilidad del procesamiento de imágenes es muy amplia y abarca muchos campos. Por ejemplo, imágenes obtenidas con fines de diagnóstico médico, imágenes aéreas obtenidas para realizar exámenes del terreno, aplicaciones en automatización industrial involucrando el uso de sensores visuales en robots, circuitos de vigilancia automatizada, etc.

La figura 14 ejemplifica las cinco etapas de procesamiento digital de imágenes descrito anteriormente.

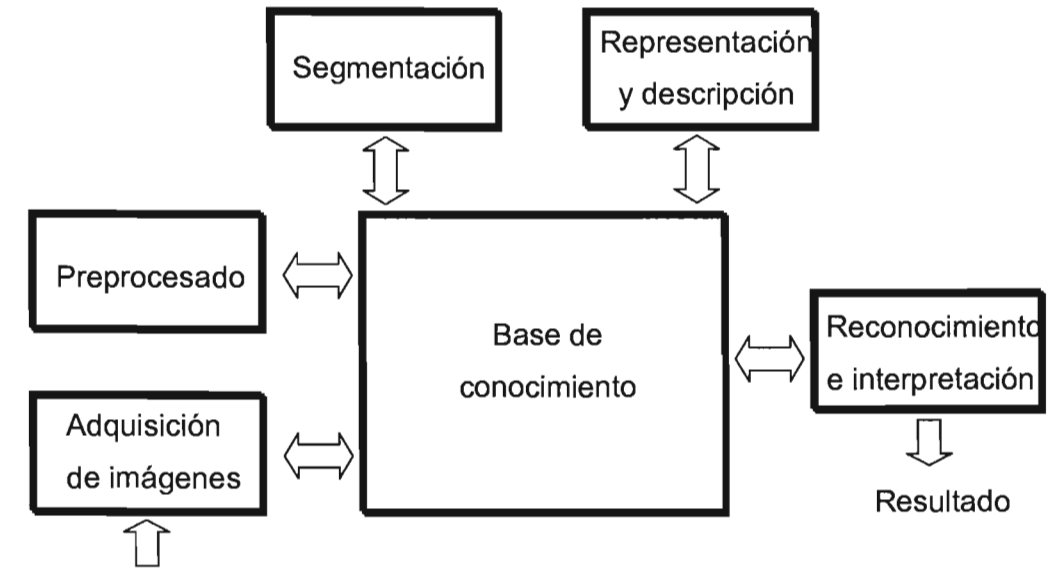


Figura 14 Etapas del procesamiento digital de imagen

II.III Modelo de imagen simple

El término imagen se refiere a una función bidimensional de la luz y la intensidad la cual queda indicada por $f(x,y)$, donde el valor o amplitud de f en las coordenadas espaciales (x,y) da la intensidad (iluminación) de la imagen en ese punto.

Entendiendo que la luz es una forma de energía, $f(x,y)$ debe de ser estrictamente mayor que cero y finita, es decir,

$$0\langle f(x,y)\rangle\infty$$

La función $f(x,y)$ puede estar caracterizada por dos componentes: la cantidad de luz incidente procedente de la fuente sobre la escena y la cantidad de luz reflejada por los objetos de dicha escena es decir, por los componentes

de iluminación y reflectancia que se indican como $i(x,y)$ y $r(x,y)$ respectivamente.

Finalmente la función $f(x,y)$ queda definida como:

$$f(x,y)=i(x,y)r(x,y)$$

donde

$$0\langle i(x,y)\langle\infty \quad y \quad 0\langle r(x,y)\langle 1$$

A la intensidad de una imagen monocromática f en las coordenadas (x,y) se le denomina nivel de gris de la imagen en ese punto.

II.IV Muestreo y cuantificación

El muestreo tiene como finalidad cambiar el dominio de la señal eléctrica del espacio continuo al espacio discreto para que en la fase de cuantización sea medida cada una de las muestras tomadas a través de una cámara, de igual forma tiene el efecto de reducir la resolución espacial de cada una de dichas muestras.

El muestreo de una señal analógica consiste en tomar muestras de su amplitud a intervalos regulares de tiempo. Estas muestras pueden tener cualquier valor numérico.

Para que las muestras sean una representación suficiente de la señal analógica, el número de muestras por segundo, es decir, la frecuencia de muestreo, será por lo menos el doble de la mayor frecuencia contenida en el espectro de la señal.

En una imagen digitalizada, el número de renglones y columnas correspondientes a la rejilla de muestreo, ocasionaran una mayor o menor definición de la imagen.

La cuantización consiste en sustituir estos infinitos valores por el número entero más próximo. La diferencia entre el valor real de la muestra y su aproximación a un entero se denomina *error de cuantificación*. Puede decirse que el muestreo discretiza la dimensión temporal de la señal analógica y que la cuantificación discretiza su rango de voltajes.

En la figura 15 se observa un ejemplo de una imagen a distintos niveles de muestreo y cuantificación.

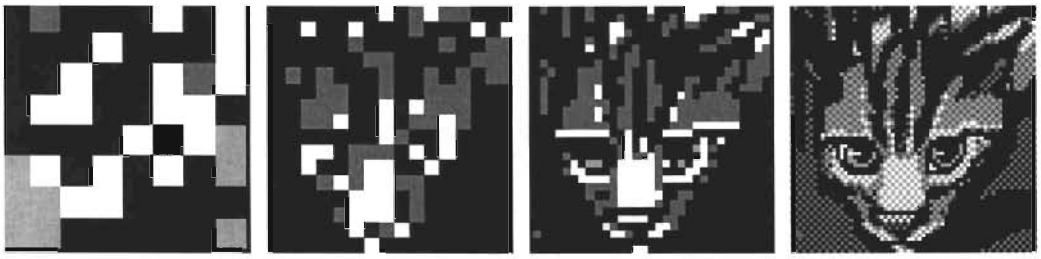


Figura 15 Ejemplo de niveles de muestreo y cuantificación en una imagen

II.V Muestreo uniforme y cuantificación

La digitalización de las coordenadas espaciales (x, y) se denomina muestreo de la imagen y la digitalización de la amplitud se conoce como cuantificación del nivel de gris.

Suponiendo que una imagen continua $f(x, y)$ se describe de forma aproximada por una serie de muestras igualmente espaciadas organizadas en forma de una matriz $N \times M$ donde cada elemento de la matriz es una cantidad discreta como se muestra a continuación:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,M-1) \\ f(1,0) & f(1,1) & \dots & f(1,M-1) \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1,M-1) \end{bmatrix}$$

El término de la derecha representa una imagen digital y cada elemento de la matriz se puede denominar elemento de la imagen, pixel o pel.

Formalizando lo anterior y suponiendo que Z y R representan a los conjuntos de números enteros y reales respectivamente, el proceso de muestreo puede entenderse como una partición en una cuadrícula del plano xy, siendo las coordenadas del centro de cada elemento de la cuadrícula un par de elementos del producto cartesiano $Z \times Z$ también indicado por Z^2. Por tanto $f(x,y)$ representa una imagen digital si (x,y) son enteros de $Z \times Z$ y f es una función que asigna un nivel de gris a cada par de coordenadas (x,y) distinto. Si los niveles de gris también son números enteros entonces Z reemplaza a R y una imagen digital se convierte en una función bidimensional $(2-D)$ cuyas coordenadas y valores de amplitud son números enteros.

II.VI Muestreo no uniforme y cuantificación

Para un valor fijo de la resolución espacial, la apariencia de una imagen puede mejorarse en muchos casos empleando un esquema adaptativo en el que el proceso de muestreo dependa de las características de la imagen. En general, se necesita un muestreo fino en las proximidades de las transiciones bruscas en los niveles de gris, mientras que se puede aplicar un muestreo tosco en las regiones relativamente suaves.

La necesidad de identificar los contornos, aunque solamente sea de forma aproximada, es uno de los inconvenientes definidos en el muestreo no uniforme.

Cuando el número de niveles de gris debe mantenerse reducido, el empleo de niveles desigualmente espaciados es a menudo deseable en el proceso de cuantificación.

II.VII Definición de Pixel

La parte más pequeña de una imagen es un *punto* que recibe el nombre de *píxel*. La palabra pixel surge de la combinación de dos palabras inglesas comunes, *pinture* (imagen) y *element* (elemento).

Un pixel se describe como una unidad lógica, y no física, ya que el tamaño físico de un pixel individual lo determina el tamaño de la imagen.

El color específico de un pixel en imágenes a color es una combinación de tres componentes del espectro de colores rojo, verde y azul (también conocido como RGB)

II.VIII Relaciones básicas entre píxeles

Como se menciono anteriormente, una imagen se indica por $f(x,y)$. Cuando se haga referencia a un píxel en particular, se emplearán letras minúsculas, como p y q, un subconjunto de pixeles de $f(x,y)$ se indicará mediante *S*.

II.IX Vecinos de un píxel

Un pixel p de coordenadas (x,y) tiene cuatro vecinos horizontales y verticales cuyas coordenadas vienen dadas por:

$$(x+1,y),(x-1,y),(x,y+1),(x,y-1)$$

Este conjunto de pixeles, denominados los *4-vecinos de* p, se representan por $N_4(p)$. Cada píxel está a una unidad de distancia de (x,y), y algunos de los vecinos de p caen fuera de la imagen digital si (x,y) está en el borde de la imagen.

Los cuatro vecinos en diagonal de p tienen las coordenadas:

$$(x+1,y+1),(x+1,y-1),(x-1,y+1),(x-1,y-1)$$

y se representan por $N_D(p)$. Estos puntos, junto con los 4-vecinos, se denominan los *8-vecinos de p* y se representan por $N_8(p)$.

En la figura 16 se representan los 4 y 8 vecinos de un pixel.

Figura 16 Esquema representativo de los 4 y 8 vecinos de un pixel

II.X Conectividad

La conectividad entre pixeles es un concepto importante empleado para establecer los límites de los objetos y los componentes de áreas en una imagen. Para determinar si dos pixeles están conectados, debe determinarse si son adyacentes en algún sentido (como ser 4 vecinos) y si sus niveles de gris cumplen un criterio especificado de similitud (como ser iguales).

Sea *V* el conjunto de valores de niveles de gris empleados para definir la conectividad. Se consideran tres tipos de conectividad:

a) 4-conectividad. Dos pixeles *p* y *q* con valores dentro de *V* están 4 conectados si *q* pertenece a $N_4(p)$.

b) 8-conectividad. Dos pixeles *p* y *q* con valores dentro de *V* están 8 conectados si *q* pertenece a $N_8(p)$.

c) m-conectividad (conectividad mixta). Dos pixeles *p* y *q* con valores dentro de *V* están m-conectados si:

   i. *q* pertenece a $N_4(p)$, o bien

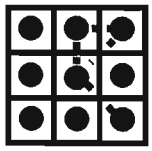
   ii. *q* pertenece a $N_D(p)$ y además el conjunto $N_4(p) \cap N_4(q)$ es vacío.

La conectividad mixta es una modificación de la 8-conectividad que se introdujo para eliminar los múltiples caminos de conexión que aparecen a menudo cuando se emplea la 8-conectividad.

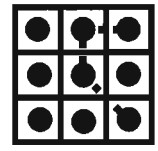
Por otro lado, un pixel *p* es *adyacente* de un pixel *q* si están conectados.

Se puede definir 4, 8 o m adyacencia, dependiendo del tipo de conectividad especificada.

La figura 17 muestra la conectividad existente entre los 8 vecinos del pixel central así como la conectividad de los *m* vecinos del pixel central.



Conectividad entre los 8 vecinos del pixel central



Conectividad entre los m vecinos del pixel central

Figura 17 Diagramas de conectividad

II.XI Medidas de distancia

Para los pixeles *p*, *q* y *z* de coordenadas $(x,y),(s,t)y(u,v)$ respectivamente, *D* es una *función distancia* o una *métrica* si:

a) $D(p,q) \geq 0 (D(p,q) = 0 \quad si\ y\ solo\ si \quad p = q)$

b) $D(p,q) = D(q,p)$

c) $D(p,z) \leq D(p,q) + D(q,z)$

La *distancia euclídea* entre *p* y *q* está definida por:

$$D_e(p,q) = \left[(x-s)^2 + (y-t)^2\right]^{\frac{1}{2}}$$

Para este medida de distancia, los pixeles que están a una distancia menor o igual que algún valor *r* de (x,y) son los puntos contenidos en un círculo de radio *r* con centro en (x,y).

La distancia D_4 denominada también distancia *city block* entre dos pixeles *p* y *q* se define como:

$$D_4(p,q) = |x-s| + |y-t|$$

En este caso los pixeles que estén a una distancia de (x,y), D_4 menor o igual que un determinado valor de r forman un rombo centrado en (x,y).

La distancia D_8 también llamada *distancia de tablero de ajedrez* entre *p* y *q* se define como:

$$D_8(p,q) = max(|x-s|,|y-t|)$$

En este caso los pixeles a una distancia D_8 de (x,y) menor o igual que un cierto valor *r* forman un cuadrado centrado en (x,y)

II.XII Operaciones aritmético-lógicas

Las operaciones aritméticas y lógicas entre dos pixeles *p* y *q* son:

a) Adición $p+q$

b) Sustracción $p-q$

c) Multiplicación $p*q$ (o también *pq* y *pxq*)

d) División $p \div q$

Las operaciones aritméticas sobre imágenes completas se realizan pixel a pixel.

Las principales operaciones lógicas empleadas en el procesamiento de imágenes son el *Y* lógico (AND), el *O* lógico (OR) y el COMPLEMENTO lógico (COMPLEMENT), representados por:

a) Y : *p* AND *q* (o también $p.q$)

b) O : *p* OR *q* (o también $p+q$)

c) COMPLEMENTO : NOT *q* (o también $\bar{q}$)

Las operaciones lógicas se aplican solamente a las imágenes binarias, mientras que las operaciones aritméticas se aplican a pixeles con varios valores.

II.XIII Histograma

El histograma es la base de una gran cantidad de técnicas de procesado de imagen, básicamente es una representación gráfica de las frecuencias relativas con las que aparecen los distintos colores en una determinada imagen.

El histograma de una imagen es un gráficos del número de pixeles en la imagen como función de la intensidad de la luz o nivel de gris.

De forma genérica, se representa como un gráfico de barras en el que las abscisas son los distintos colores de la imagen y las ordenadas la frecuencia relativa con la que cada color aparece en la imagen.

El histograma proporciona información sobre el brillo y el contraste de la imagen y puede ser utilizado para ajustar dichos parámetros, eliminar ciertas tonalidades, etc.

La figura 18 muestra el histograma obtenido del análisis de una imagen.

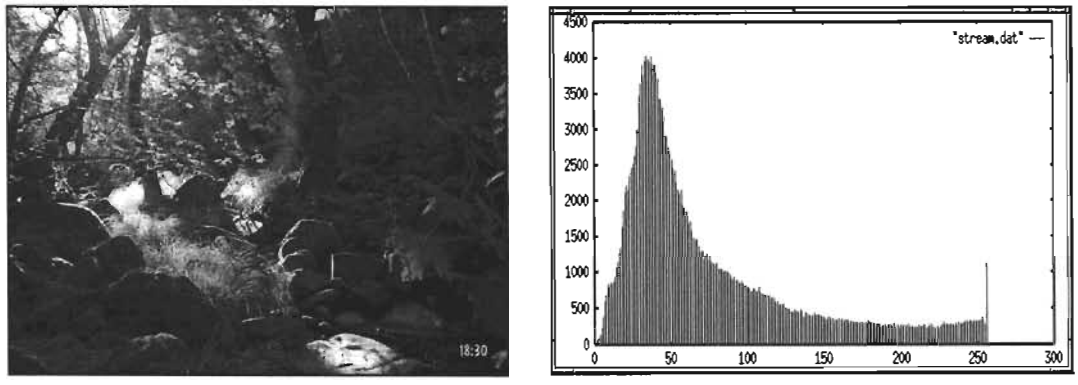


Figura 18 Histograma de una imagen.

CAPÍTULO III

PROCESAMIENTO DIGITAL DE IMÁGENES

CAPÍTULO III

En los últimos años, el *procesamiento digital de imágenes* ha sido utilizado por diversas disciplinas tales como la medicina, biología, física e ingeniería para llevar a cabo procesos que requieren de un tratamiento especial en las imágenes recolectadas y procesadas a través de sistemas de computo.

Mediante el procesamiento digital de imágenes es posible manipular las mismas con el fin de obtener información objetiva de una escena captada por medio de una cámara.

Dos de las tareas fundamentales por las cuales se acentúa la importancia de este tipo de procesamiento son el mejoramiento de la imagen digital con fines interpretativos y la toma de decisiones de manera automática de acuerdo al contenido de la imagen digital.

III.I Segmentación de imágenes

El primer paso del análisis de imágenes consiste generalmente en segmentar la imagen.

La segmentación subdivide una imagen en sus partes constituyentes u objetos, es decir, la segmentación de imágenes consiste en dividir o separar aquellas partes de la imagen que pertenezcan a una misma estructura El nivel al que se lleva a cabo esta subdivisión depende del problema a resolver. Esto es, la subdivisión deberá detenerse cuando los objetos de interés de una aplicación hayan sido aislados.

En general, la segmentación autónoma es una de las tareas más difíciles del procesamiento de imágenes.

Los algoritmos de segmentación de imágenes monocromáticas generalmente se basan en una de las dos propiedades básicas de los valores de nivel de gris: discontinuidad y similaridad. En la primera categoría el método consiste en dividir una imagen basándose en los cambios bruscos de nivel de gris. Las principales áreas de interés de esta categoría son la detección de puntos aislados y la detección de líneas y bordes de una imagen. Los

principales métodos de la segunda categoría están basados en la umbralización, crecimiento de región, división y fusión de regiones. El concepto de segmentación de una imagen basado en la discontinuidad o similaridad de los valores de nivel de gris de sus pixeles es aplicable tanto a las imágenes estáticas como a las dinámicas (variantes en el tiempo).

La figura 19 muestra la imagen de una célula después de haber pasado por el proceso de segmentación.

Figura 19 Imagen segmentada

III.II Detección de bordes

Un problema de importancia fundamental en el análisis de imagen es la detección de bordes. Los contornos caracterizan las fronteras de los objetos, y por tanto son de gran utilidad de cara a la segmentación e identificación de objetos.

Los puntos de contorno son como zonas de pixeles en las que existe un cambio brusco de nivel de gris.

Una forma de realizar la detección de bordes en una imagen es utilizando derivadas de la misma. La derivada es una medida de cambio de una función respecto a una variable. Los bordes radican en donde dichos cambios son máximos.

$$\max\left[\left(\frac{\partial f(x,y)}{\partial x}\right)^2+\left(\frac{\partial f(x,y)}{\partial y}\right)^2\right]^{\frac{1}{2}}$$

En el tratamiento de imágenes se trabaja con pixeles y en un ambiente discreto por lo que a través del tiempo se han logrado desarrollar diferentes operadores que permiten detectar contornos de imágenes basados en *máscaras* que representan aproximaciones en diferencias finitas de los gradientes ortogonales $\left(\frac{\partial f(x,y)}{\partial x}\right)^2$ y $\left(\frac{\partial f(x,y)}{\partial y}\right)^2$.

La tabla (c) muestra los operadores gradientes más comúnmente utilizados en el campos del procesamiento digital de imágenes.

Operadores Gradiente	**Dirección horizontal**	**Dirección vertical**
Roberts	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$
Smoothed (Prewitt)	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$
Sobel	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
Isotrópico	$\begin{bmatrix} -1 & 0 & 1 \\ -\sqrt{2} & 0 & \sqrt{2} \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix}$

Tabla (c) Operadores gradientes más utilizados.

La figura 20 muestra una imagen después de haberle aplicado un operador gradiente (Sobel) para detección de bordes.

Figura 20 Detección de bordes

III.III Detección de discontinuidades

Existen tres tipos básicos de discontinuidades de una imagen digital: puntos, líneas y bordes.

La forma más común de ver discontinuidades es pasar una máscara a través de la imagen. Para una máscara de 3×3 como la mostrada en la figura 21 el procedimiento implica calcular la suma de los productos de los coeficientes por los niveles de gris contenidos en la región encerrada por las máscara. Esto es, la respuesta de la máscara en un punto cualquiera de la imagen es:

$$R = w_1 z_1 + w_2 z_2 + \ldots + w_9 z_9 = \sum_{i=1}^{9} w_i z_i$$

donde z_i es el nivel de gris asociado con el coeficiente de la máscara w_i.

W_1	W_2	W_3
W_4	W_5	W_6
W_7	W_8	W_9

Figura 21 Máscara general de 3 X 3

III.IV Enlazado de bordes y detección de límites

Idealmente, las técnicas que detectan discontinuidades de intensidad en una imagen digital deberían obtener solamente pixeles situados en el límite entre regiones. En la práctica, este conjunto de pixeles rara vez caracteriza completamente un límite debido al ruido, a las interrupciones en el límite por una iluminación no uniforme, así como a otros efectos que introducen discontinuidades de intensidad. Por ello, los algoritmos de detección de bordes normalmente se continúan por procedimientos de enlazado y de detección de límites diseñados para reunir pixeles del borde en límites que tengan algún sentido.

III.V Umbralización

La umbralización es uno de los métodos más importantes en la segmentación de imágenes, su objetivo principal es convertir una imagen con varios niveles de gris es una imagen monocromática (blanco y negro) que tenga la información esencial relativa al número, posición, tamaño y forma de los objetos que se encuentran en una imagen.

Una forma simple de seleccionar un umbral es calculando la media aritmética de los datos disponibles:

$$T = \tilde{x}(i)$$

Esta selección del umbral equivale a fijar el porcentaje de muestras por debajo del umbral en un 50% aproximadamente.

A esta técnica de umbralización se le conoce como *umbralización utilizando la media aritmética.*

La figura 22 muestra una imagen en la que se ha utilizado la técnica de umbralización utilizando la media aritmética.

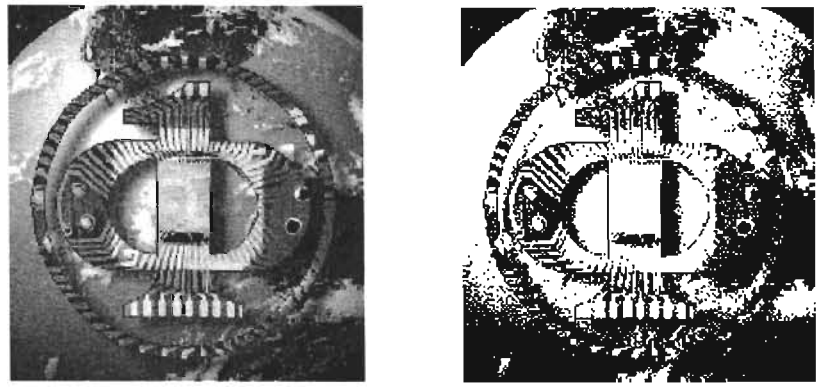


Figura 22 Umbralización utilizando la media aritmética del histograma.

Otro tipo de umbralización es la multinivel la cual presenta dificultades al establecer umbrales múltiples que aíslen efectivamente las regiones de interés.

La umbralización se puede contemplar como una operación que implica realizar comprobaciones frente a una función T de la forma:

$$T = T[x, y, p(x, y), f(x, y)]$$

donde $f(x,y)$ es el nivel de gris del punto (x,y), y $p(x,y)$ representa alguna propiedad local de este punto, por ejemplo, la medida del nivel de gris de una vecindad centrada en (x,y). Una imagen umbralizada $g(x,y)$ se define como:

$$g(x,y) = \begin{cases} 1 \; si \; f(x,y) \rangle T \\ 0 \; si \; f(x,y) \leq T \end{cases}$$

De este modo los pixeles marcados con 1 (o con cualquier otro nivel de intensidad conveniente) corresponden a objetos, mientras que los píxeles marcados con 0 corresponden al fondo.

Cuando T depende solamente de $f(x,y)$, el umbral se denomina *global.*

La más sencilla de todas las técnicas de umbralización es la partición del histograma de una imagen utilizando un umbral único, T.

Lo anterior utiliza sólo una variable de intensidad.

En algunos casos, se utilizará más de una variable para caracterizar a cada pixel de una imagen. Un ejemplo de lo anterior está en las imágenes en color, donde se utilizan el componente rojo *R*, verde *G*, y azul *B* para formar una imagen de color compuesta. En este caso, cada pixel está caracterizado por tres valores, lo que permite la construcción de un histograma tridimensional.

La umbralización se concentra ahora en encontrar agrupaciones de puntos en un espacio tridimensional.

III.VI Segmentación orientada a regiones

El objetivo de la segmentación es dividir una imagen en regiones.

Sea R la representación de la región completa de una imagen. Se puede contemplar la segmentación como un proceso que divide a R en n subregiones, $R_1, R_2, \ldots, R_n$ de tal forma que:

a) $\bigcup_{i=1}^{n} R_i = R$

b) R_i *es una región conexa,* $i = 1,2,\ldots,n$

c) $R_i \cap R_j = \varnothing$ *para todo* i *y* $j, i \neq j$

d) $P(R_i) =$ *verdadero para* $i = 1,2,\ldots,n$

e) $P(R_i \cup R_j) =$ *falso para* $i \neq j$

La condición a) indica que la segmentación debe ser completa, es decir, cada pixel debe estar en una región. La segunda condición requiere que los puntos de una región deban ser conexos. La condición c) indica que las regiones deben ser disjuntas. La condición d) trata las propiedades que deben satisfacer los pixeles de una región segmentada. La condición e) indica que las regiones R_i y R_j son diferentes en el sentido del predicado P.

El *crecimiento de regiones* es un procedimiento que agrupa pixeles o subregiones dentro de regiones más grandes. Este método implica el agregar pixeles que comienzan con un conjunto de puntos *generadores* a partir de los que van creciendo en las regiones al agregar a cada uno de estos puntos los pixeles próximos que tienen propiedades similares (como nivel de gris, textura o color).

Los dos problemas inmediatos derivados de este procedimiento, son la selección de los generadores iniciales que representen correctamente a las regiones de interés y la selección de las propiedades adecuadas para la inclusión de puntos en las diversas regiones durante el proceso de crecimiento.

Otra alternativa consiste en subdividir la imagen inicialmente en un conjunto de regiones arbitrarias disjuntas y después fusionar y/o dividir las regiones intentando satisfacer las condiciones establecidas anteriormente.

III.VII Utilización del movimiento en la segmentación

El movimiento es una poderosa herramienta utilizada por los seres humanos y los animales para extraer objetos de interés de un fondo de detalles irrelevantes. En las aplicaciones de imágenes, el movimiento se deriva de un desplazamiento relativo entre el sistema de sensores y la escena que se está presentando, como en el caso de análisis dinámico de sistemas.

La diferencia entre dos imágenes en un problema de imágenes dinámicas tiende a eliminar todos los componentes estacionarios, dejando solamente los elementos de la imagen que corresponden al ruido y a los objetos en movimiento. El problema del ruido se puede tratar por métodos de filtrado o por la formación de una imagen de diferencia acumulativa.

En la práctica, no siempre es posible la obtención de una imagen de referencia solamente con elementos estacionarios, y se hace necesaria la construcción de una a partir de un conjunto de imágenes que contienen uno o más objetos en movimiento.

Un procedimiento para generar una imagen de referencia es primeramente considerar que la primera imagen de una sucesión es la de referencia.

Cuando un componente no estacionario se ha movido completamente fuera de su posición en el fotograma de referencia, el fondo correspondiente del fotograma actual puede duplicarse en la posición ocupada originalmente por el objeto en el fotograma de referencia. Cuando todos los objetos en movimiento se han desplazado completamente fuera de sus posiciones originales, una imagen de referencia contendrá solamente los componentes estacionarios que se han ido creando.

III.VIII Ejemplo de aplicación con procesamiento digital de imagen

El proceso por el cual atraviesa una imagen al ser procesada mediante un operador es a grandes rasgos el mismo. Se tiene la imagen original, se obtienen los valores de los píxeles que la componen y se aplica el operador mediante una *máscara* dando como resultado nuevos valores en los pixeles en donde fue aplicada la máscara y por lo tanto una imagen procesada.

El tratamiento descrito anteriormente se muestra en la figura 23.

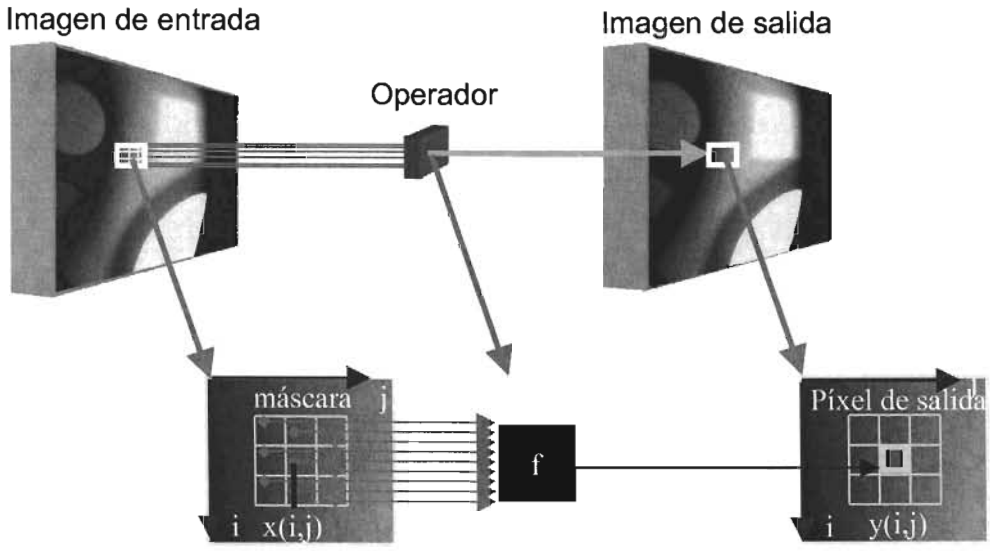


Figura 23 Aplicación de un operador sobre una imagen

Es decir, suponiendo que se tiene una imagen monocromática con las siguientes características.

25	85	89	96	58
45	82	74	58	125
14	69	24	125	123
17	15	19	25	23
11	14	52	56	85

Tomando al operador *Sobel*.

G_x

-1	-2	-1
0	0	0
1	2	1

G_y

-1	0	1
-2	0	2
-1	0	1

Basando los cálculos en las ecuaciones

$$G_X = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$
$$G_Y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

Sustituyendo en las ecuaciones anteriores los valores correspondientes a los pixeles de la imagen original para obtener los valores de la nueva imagen.

$$G_{11} = (-1)*(0)+(-2)*(0)+(-1)*(0)+(0)*(0)+(0)*(25)+(0)*(85)+(1)*(0)+(2)*(45)+(1)*(82)$$
$$G_{11} = 172$$
$$G_{21} = (-1)*(0)+(-2)*(0)+(-1)*(0)+(0)*(0)+(0)*(25)+(0)*(85)+(1)*(45)+(2)*(82)+(1)*(74)$$
$$G_{21} = 283 = 255$$
$$G_{31} = (-1)*(0)+(-2)*(0)+(-1)*(0)+(0)*(0)+(0)*(25)+(0)*(85)+(1)*(82)+(2)*(74)+(1)*(58)$$
$$G_{31} = 288 = 255$$
$$G_{41} = (-1)*(0)+(-2)*(0)+(-1)*(0)+(0)*(0)+(0)*(25)+(0)*(85)+(1)*(74)+(2)*(58)+(1)*(125)$$
$$G_{41} = 315 = 255$$
$$G_{51} = (-1)*(0)+(-2)*(0)+(-1)*(0)+(0)*(0)+(0)*(25)+(0)*(85)+(1)*(58)+(2)*(125)+(1)*(0)$$
$$G_{51} = 308 = 255$$

$$G_{12} = (-1)*(0)+(-2)*(25)+(-1)*(85)+(0)*(0)+(0)*(45)+(0)*(82)+(1)*(0)+(2)*(14)+(1)*(69)$$
$$G_{12} = -38 = 0$$
$$G_{22} = (-1)*(25)+(-2)*(85)+(-1)*(89)+(0)*(45)+(0)*(82)+(0)*(74)+(1)*(14)+(2)*(69)+(1)*(241)$$
$$G_{22} = 109$$
$$G_{32} = (-1)*(85)+(-2)*(89)+(-1)*(96)+(0)*(82)+(0)*(74)+(0)*(58)+(1)*(69)+(2)*(24)+(1)*(125)$$
$$G_{32} = -117 = 0$$
$$G_{42} = (-1)*(89)+(-2)*(96)+(-1)*(58)+(0)*(74)+(0)*(58)+(0)*(125)+(1)*(241)+(2)*(125)+(1)*(123)$$
$$G_{42} = 266 = 255$$
$$G_{52} = (-1)*(96)+(-2)*(58)+(-1)*(0)+(0)*(58)+(0)*(125)+(0)*(0)+(1)*(125)+(2)*(123)+(1)*(0)$$
$$G_{52} = 159$$

$$G_{13} = (-1)*(0)+(-2)*(14)+(-1)*(69)+(0)*(0)+(0)*(147)+(0)*(158)+(1)*(0)+(2)*(112)+(1)*(114)$$
$$G_{13} = 141$$
$$G_{23} = (-1)*(14)+(-2)*(69)+(-1)*(24)+(0)*(17)+(0)*(15)+(0)*(19)+(1)*(11)+(2)*(14)+(1)*(52)$$
$$G_{23} = -85 = 0$$
$$G_{33} = (-1)*(69)+(-2)*(24)+(-1)*(125)+(0)*(15)+(0)*(19)+(0)*(25)+(1)*(14)+(2)*(52)+(1)*(56)$$
$$G_{33} = -68 = 0$$
$$G_{43} = (-1)*(24)+(-2)*(125)+(-1)*(123)+(0)*(19)+(0)*(25)+(0)*(23)+(1)*(52)+(2)*(56)+(1)*(85)$$
$$G_{43} = -148 = 0$$
$$G_{53} = (-1)*(125)+(-2)*(123)+(-1)*(0)+(0)*(25)+(0)*(23)+(0)*(0)+(1)*(56)+(2)*(85)+(1)*(0)$$
$$G_{52} = -45 = 0$$

$$G_{14} = (-1)*(0)+(-2)*(17)+(-1)*(15)+(0)*(0)+(0)*(11)+(0)*(14)+(1)*(0)+(2)*(0)+(1)*(0)$$
$$G_{14} = -49 = 0$$
$$G_{24} = (-1)*(25)+(-2)*(85)+(-1)*(89)+(0)*(45)+(0)*(82)+(0)*(74)+(1)*(14)+(2)*(69)+(1)*(241)$$
$$G_{24} = 183$$
$$G_{34} = (-1)*(85)+(-2)*(89)+(-1)*(96)+(0)*(82)+(0)*(74)+(0)*(58)+(1)*(69)+(2)*(241)+(1)*(125)$$
$$G_{34} = 317 = 255$$
$$G_{44} = (-1)*(89)+(-2)*(96)+(-1)*(58)+(0)*(74)+(0)*(58)+(0)*(125)+(1)*(241)+(2)*(125)+(1)*(123)$$
$$G_{44} = 271 = 255$$
$$G_{54} = (-1)*(96)+(-2)*(58)+(-1)*(0)+(0)*(58)+(0)*(125)+(0)*(0)+(1)*(125)+(2)*(123)+(1)*(0)$$
$$G_{54} = 159$$

$$G_{15} = (-1)*(0)+(-2)*(25)+(-1)*(85)+(0)*(0)+(0)*(45)+(0)*(82)+(1)*(0)+(2)*(14)+(1)*(69)$$
$$G_{15} = -38 = 0$$
$$G_{25} = (-1)*(25)+(-2)*(85)+(-1)*(89)+(0)*(45)+(0)*(82)+(0)*(74)+(1)*(14)+(2)*(69)+(1)*(241)$$
$$G_{25} = 109$$
$$G_{35} = (-1)*(85)+(-2)*(89)+(-1)*(96)+(0)*(82)+(0)*(74)+(0)*(58)+(1)*(69)+(2)*(241)+(1)*(125)$$
$$G_{35} = 317 = 255$$
$$G_{45} = (-1)*(89)+(-2)*(96)+(-1)*(58)+(0)*(74)+(0)*(58)+(0)*(125)+(1)*(241)+(2)*(125)+(1)*(123)$$
$$G_{45} = 271 = 255$$
$$G_{55} = (-1)*(96)+(-2)*(58)+(-1)*(0)+(0)*(58)+(0)*(125)+(0)*(0)+(1)*(125)+(2)*(123)+(1)*(0)$$
$$G_{55} = 159$$

La imagen monocromática resultante una vez aplicado el operador Sobel en el eje x y y es:

172	255	255	255	255
0	109	0	255	159
141	255	0	0	0
0	183	255	255	159
0	109	0	255	159

De forma gráfica se muestra en la figura 24 una imagen monocromática con su respectivo histograma de los valores de píxeles que la componen. Así mismo, se pueden observar los cambios realizados en la misma imagen una vez que ha sido procesada mediante un operador de tipo Sobel en el eje de las *X* y el eje de las *Y*.

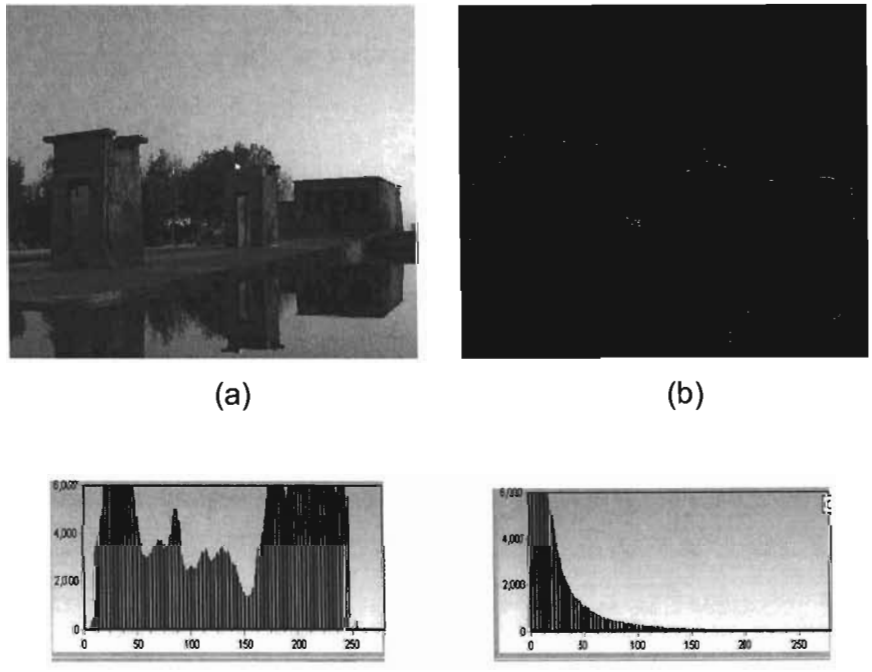


Figura 24 Imagen procesada mediante un operador de tipo Sobel. (a) imagen original, (b) imagen procesada mediante el operador Sobel, (c) histograma de la imagen original, (d) histograma de la imagen procesada.

CAPÍTULO IV

FILTROS PREDICTIVOS

CAPÍTULO IV

La utilización de filtros en el tratamiento digital de imágenes ha tomado una especial importancia en los últimos años. Investigaciones realizadas en este campo, han incrementado considerablemente el uso de filtros en diferentes procesos de análisis, lo anterior porque por medio de ellos es posible resaltar los bordes internos de una imagen, eliminar el ruido que pueda tener, etc.

Los filtros se pueden identificar como complejos algoritmos que se aplican a los pixeles de una imagen para modificarlos es decir; son cálculos matemáticos aplicados a los pixeles de una imagen.

IV.I Filtros Predictivos

Los *filtros predictivos* son por si mismos sistemas que pueden "auto ajustarse" para responder a cambios en su entorno, lo que implica que los parámetros que caracterizan al filtro son cambiantes con el tiempo. Por lo anterior, se les puede dar una aplicación en diferentes campos de investigación tales como comunicaciones, ingeniería biomédica, sistemas expertos, etc.

Existen cuatro fases que conforman la implementación de los filtros predictivos.

- Identificación. Reconocimiento del objeto o señal a analizar.
- Modelación inversa. Obtención de los parámetros o variables iniciales.
- Predicción. Mediante ecuaciones matemáticas, predecir el posible comportamiento del objeto o señal antes de que este cambie.
- Cancelación de interferencia. Capacidad de abortar el análisis dado un error en el proceso mediante condiciones externas al filtro.

IV.II Filtros g-h y g-h-k

- Filtro g-h

El filtro g-h es un tipo de filtro predictivo lo que implica que primeramente se deben de saber determinar las condiciones iniciales del objeto a analizar para posteriormente poder predecir la ubicación (posición y velocidad) de este, es decir el filtro g-h puede adelantar la predicción de un tiempo n a un tiempo $n+1$ después de haber realizado una primera observación. Para una mejor comprensión de lo anterior, se hará uso a continuación de un ejemplo.

Considerando que se tienen la posición y velocidad de un objeto en un tiempo $n-1$ y que el objeto puede tener una velocidad en un tiempo $n-1$ de $200\frac{m}{s}$, con un periodo T entre cada observación de $10s$, se establecen las condiciones iniciales para el análisis.

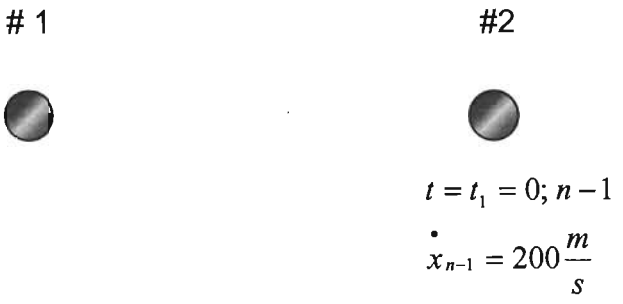
Dado lo anterior se determina que el objeto está $\left(200\frac{m}{s}\right)(10s)=2000m$ más lejos del punto de origen en un tiempo *n* que en un tiempo $n-1$. Esta será la posición x_n.

De igual forma, considerando que en un tiempo n el objeto está en una posición a una distancia de $60m$ más lejos del punto de inicio; y que el rango de medición del objeto con el cual se están realizando las observaciones es de $0.1m$ entonces, la velocidad actual del objeto será:

$$Velocidad\ \ Actual = 200\frac{m}{s}+\left(\frac{60m}{10s}\right)=206\frac{m}{s} \quad \text{.....(a)}$$

El comportamiento descrito anteriormente se encuentra ejemplificado en la figura 25.

Primera observación



Segunda observación $t = t + T$

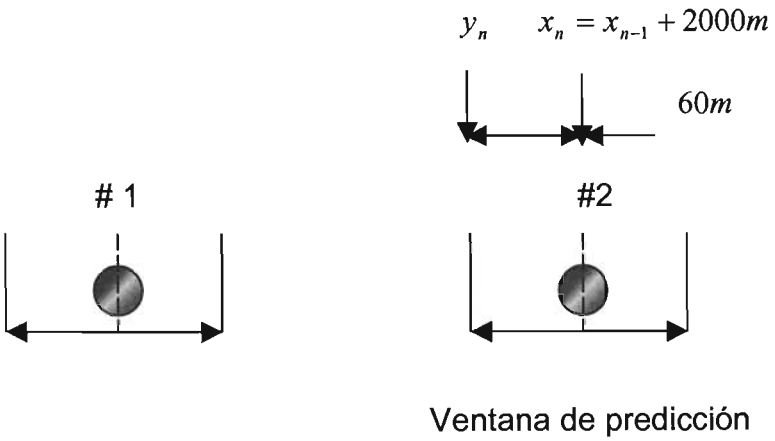


Figura 25 Predicción de la posición x_n y y_n

Utilizando la fracción $\frac{1}{10}$ del incremento en velocidad de $6\frac{m}{s}$. La velocidad actual estará dada por:

$$Velocidad\ \ Actual = 200\frac{m}{s} + \frac{1}{10}\left(\frac{60m}{10s}\right) = 200\frac{m}{s} + 0.60\frac{m}{s} = 200.6\frac{m}{s} \quad .....(b)$$

En búsquedas posteriores si la velocidad del objeto se incrementa $0.6\frac{m}{s}$ en promedio, después de 10 observaciones la velocidad del objeto estará incrementada $6\frac{m}{s}$ y se tendría la velocidad correcta. De otra manera, si la velocidad del objeto es de $200\frac{m}{s}$, entonces en observaciones sucesivas la medición de la posición del objeto sería igualmente probable antes o después de la posición predicha, así que en promedio la velocidad del objeto no sería diferente de un valor inicial estimado de $200\frac{m}{s}$.

Utilizando la ecuación (b) en forma paramétrica se obtiene:

$$\dot{x}_n = \dot{x}_n + h_n\left(\frac{y_n - x_n}{T}\right) \quad \text{.....(c)}$$

La fracción $\frac{1}{10}$ está representada por el parámetro h_n. El prefijo n es usado para indicar que en general el parámetro h dependerá del tiempo. Cabe añadir que la ecuación anterior tiene un problema, el símbolo de la velocidad actual estimada después de las mediciones en el tiempo n es el mismo que el símbolo de la velocidad estimada en el tiempo n sólo antes de hacer las mediciones, ambas utilizan la variable $\dot{x}_n$. Para distinguir estas dos estimaciones, una segunda variable es añadida. Esta segunda variable indica el tiempo en el cual la última medición fue hecha para usarla estimando la velocidad del objeto.

De esta manera la ecuación (c) queda expresada como:

$$\dot{x}^*_{n,n} = \dot{x}^*_{n,n-1} + h_n\left(\frac{y_n - x^*_{n,n-1}}{T}\right) \quad \text{.....(d)}$$

La segunda variable $n-1$, de la velocidad estimada $\dot{x}^*_{n,n-1}$ indica una estimación de la velocidad del objeto en un tiempo n basado en mediciones hechas en un tiempo $n-1$ y anteriores. La segunda variable n de la velocidad estimada $\dot{x}^*_{n,n}$ viene antes del signo igual (=) que indica que esta velocidad

estimada utiliza el rango de mediciones hechos en un tiempo n, esto es, la medición del rango y_n. La variable asterisco $(*)$ indica que el parámetro es un estimado, sin embargo representa los verdaderos valores de la velocidad y posición del objeto. La figura 26 muestra un diagrama de la velocidad y posición del objeto con la nueva notación.

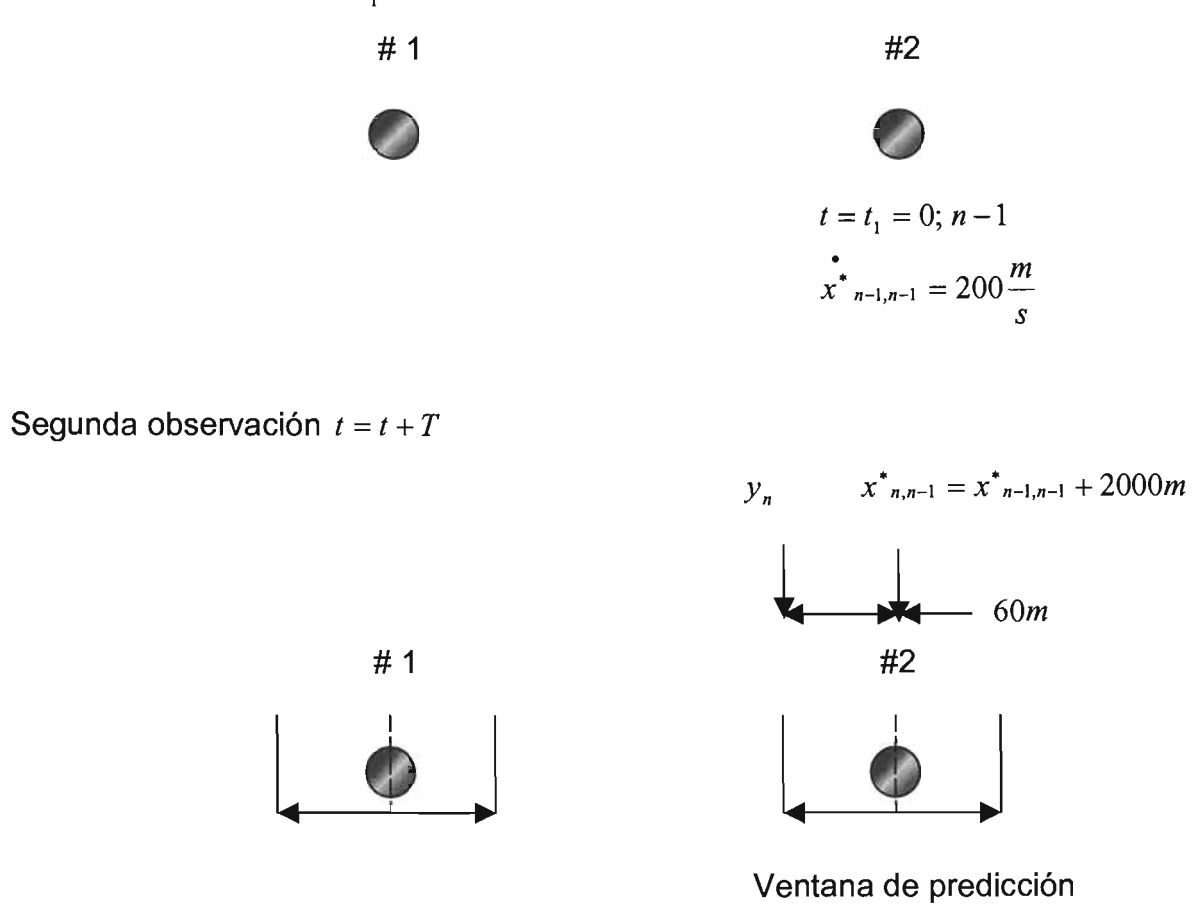


Figura 26 Predicción del objeto filtrado y medición de la posición

Para obtener la ecuación para actualizar la posición del objeto nuevamente se asume que en el tiempo $n-1$ el objeto está en un rango de $10m$ de distancia y en un tiempo *n*, $T=10s$ más tarde, el objeto con una velocidad radial de $200\frac{m}{s}$ está en un rango de $2000m$ más lejos, considerando de igual forma que al tiempo n el objeto estará $60m$ más alejado del lugar inicial.

De esta manera, la ecuación para evaluar la posición del objeto es:

$$Posición\ \ Actual = 10m + 2000m + 60m \quad \text{.....(e)}$$

Asumiendo que el objeto está $\frac{1}{6}$ de $60m$, o $10m$ más alejado del rango establecido de distancia, entonces el rango de posición actual después de las mediciones en el tiempo n estará dada por:

$$Posición\ \ Actual = 10m + 2000m + \frac{1}{6}(60m) \quad \text{.....(f)}$$

Estableciendo la ecuación (f) en forma paramétrica:

$$x^*_{n,n} = x^*_{n,n-1} + g_n\left(y_n - x^*_{n,n-1}\right) \quad \text{.....(g)}$$

Donde la fracción $\frac{1}{6}$ está representada por el parámetro g_n, el cual depende de n.

La ecuación (g) representa la ecuación deseada para la velocidad actual del objeto.

Utilizando las ecuaciones (d) y (g) se obtendrán las ecuaciones para la velocidad y posición actuales del objeto al tiempo n después que las mediciones del rango y_n del objeto hayan sido hechas.

Las ecuaciones (d) y (g) representan en conjunto las ecuaciones características de posición y velocidad del *filtro g-h*.

$$\dot{x}^*_{n,n} = \dot{x}^*_{n,n-1} + h_n\left(\frac{y_n - x^*_{n,n-1}}{T}\right) \quad \text{.....(h.1)}$$

$$x^*_{n,n} = x^*_{n,n-1} + g_n\left(y_n - x^*_{n,n-1}\right) \quad \text{.....(h.2)}$$

Estas ecuaciones proporcionan una estimación de la velocidad y posición actual del objeto, ambas basadas en mediciones presentes del rango del objeto. Son conocidas como *ecuaciones del filtro*.

La estimación $x^*_{n,n}$ es llamada *estimación del filtro*. Una estimación de x_n al tiempo presente basado en el uso de mediciones presentes y_n tan buenas como las mediciones pasadas.

Esta estimación es un contraste de la predicción estimada $x_{n,n-1}$, la cual es un estimado de x_n, basada en mediciones pasadas.

Un signo sobre la variable x es utilizado para indicar que x es la estimación predicha $x^*_{n,n-1}$, mientras una barra arriba de x es usada para indicar que x es la estimación del filtro $x^*_{n,n}$. Entonces las ecuaciones de seguimiento actual para el *filtro g-h* (h.1) y (h.2) son respectivamente:

$$\bar{\dot{x}}_n = \hat{\dot{x}}_n + h_n\left(\frac{y_n - \hat{x}_n}{T}\right) \quad \text{.....(i.1)}$$

$$\bar{x}_n = \hat{x}_n + g_n\left(y_n - \hat{x}_n\right) \quad \text{.....(i.2)}$$

Con lo anterior, se puede predecir la posición y velocidad del objeto al tiempo $n+1$ y repetir el proceso completo de velocidad y posición al tiempo $n+1$ después de que la medición y_{n+1} al tiempo $n+1$ haya sido realizada.

Para este propósito se reescriben las ecuaciones usando la nueva notación conocida como la *ecuación de transición g-h* o *ecuación de predicción*:

$$\dot{x}^*_{n+1,n} = \dot{x}^*_{n,n} \quad \text{.....(j.1)}$$

$$x^*_{n+1,n} = x^*_{n,n} + T\dot{x}^*_{n+1,n} \quad \text{.....(j.2)}$$

Estas ecuaciones permiten cambiar de la velocidad y posición al tiempo *n* a la velocidad y posición en el tiempo $n+1$ y son llamadas *ecuaciones de transición.*

En la ecuación (j.1) la estimación de la velocidad en el tiempo $n+1$, $\dot{x}^*_{n+1,n}$, es equivalente al valor $\dot{x}^*_{n,n}$ en el tiempo n, porque un modelo del objeto a velocidad constante es asumido.

- Filtro g-h-k

El filtro g-h utiliza un modelo en donde la trayectoria del objeto está caracterizada por la velocidad constante del mismo.

Considerando que pocas veces esto ocurre, ahora se analizará el caso en el que el objeto tiene una aceleración constante.

Las ecuaciones de dicho objeto en movimiento estarán dadas por:

$$x_{n+1} = x_n + \dot{x}_n T + \ddot{x}_n \frac{T^2}{2} \quad \text{.....(a.1)}$$

$$\dot{x}_{n+1} = \dot{x}_n + \ddot{x}_n T \quad \text{.....(a.2)}$$

$$\ddot{x}_{n+1} = \ddot{x}_n \quad \text{.....(a.3)}$$

Debido a las características planteadas, si se utilizan las ecuaciones dadas por el filtro g-h el resultado no sería el esperado y mucho menos el apropiado, por decirlo de alguna forma, dichas ecuaciones necesitan ser actualizadas.

El filtro g-h-k actualiza las ecuaciones correspondientes, por lo cual se tiene:

$$\ddot{x}^*_{n,n} = \ddot{x}^*_{n,n-1} + \frac{2K}{T^2}\left(y_n - x^*_{n,n-1}\right) \quad \text{.....(b.1)}$$

$$\dot{x}^*_{n,n} = \dot{x}^*_{n,n-1} + \frac{h}{T}\left(y_n - x^*_{n,n-1}\right) \quad \text{.....(b.2)}$$

$$x^*_{n,n} = \ddot{x}^*_{n,n-1} + \frac{2K}{T^2}\left(y_n - x^*_{n,n-1}\right) \quad \text{.....(b.3)}$$

De donde las ecuaciones de predicción o de transición del *filtro g-h-k* serán:

$$\ddot{x}^*_{n+,n1} = \ddot{x}^*_{n,n} \quad \text{.....(a.1)}$$

$$\ddot{x}^*_{n+,n1} = \dot{x}^*_{n,n} + \ddot{x}^*_{n,n} T \quad \text{.....(a.1)}$$

$$\ddot{x}^*_{n+,n1} = \dot{x}^*_{n,n} + \ddot{x}^*_{n,n} T + \ddot{x}^*_{n.n} \frac{T^2}{2} \quad \text{.....(a.1)}$$

IV.III Filtros α y β

Entre los filtros comúnmente utilizados para el rastreo o seguimiento de objetos se encuentra el conocido como *filtro* $\alpha\ \beta$.

El *filtro* $\alpha\ \beta$ es una variante del *filtro g-h* por lo que el desarrollo de las ecuaciones de éste último filtro da como resultado las ecuaciones que caracterizan al *filtro* $\alpha\ \beta$.

Dichas ecuaciones son:

$$\dot{x}^*_{n,n} = \dot{x}^*_{n,n-1} + \beta_n \left(\frac{y_n - x^*_{n,n-1}}{T} \right) \quad \text{.....(a)}$$

$$x^*_{n,n} = x^*_{n,n-1} + \alpha_n \left(y_n - x^*_{n,n-1} \right) \quad \text{.....(b)}$$

De donde la ecuación (a) permite calcular la velocidad del objeto y la ecuación (b) la posición del mismo.

IV.IV Filtro Kalman

- Antecedentes

En 1960 Rudolph.E. Kalman publicó su famoso artículo describiendo una solución recursiva al problema del filtrado lineal de datos discretos.

La derivación de Kalman se desarrolló dentro de un amplio contexto de modelos estado-espacio, en donde el núcleo fue la estimación por mínimos cuadrados recursivos.

Desde ese momento, el *filtro de Kalman* se ha visto sujeto a una extenso proceso de investigación y a diversas formas de aplicación, particularmente en el área de investigación autónoma y asistida, rastreo de misiles y economía.

El *Filtro Kalman* proporciona un marco para la estimación incremental de una cantidad en una situación en la cual las mediciones relacionadas con la misma están disponibles a lo largo del tiempo. Concretamente, se trata de una técnica de estimación Bayesiana empleada para seguir sistemas estocásticos dinámicos observados mediante sensores ruidosos.

Por lo anterior, el *Filtro Kalman* es el núcleo de este trabajo de tesis; utilizado junto con la técnica de *procesamiento de imágenes* permitirá estimar la posición de un objeto a través del tiempo, de tal forma que en conjunto formarán un modelo predictivo se seguimiento del objeto.

El *Filtro Kalman* es un algoritmo recursivo de procesamiento de datos, permite la estimación de datos pasados, presentes y futuros.

La figura 27 muestra un esquema del funcionamiento del F*iltro Kalman*:

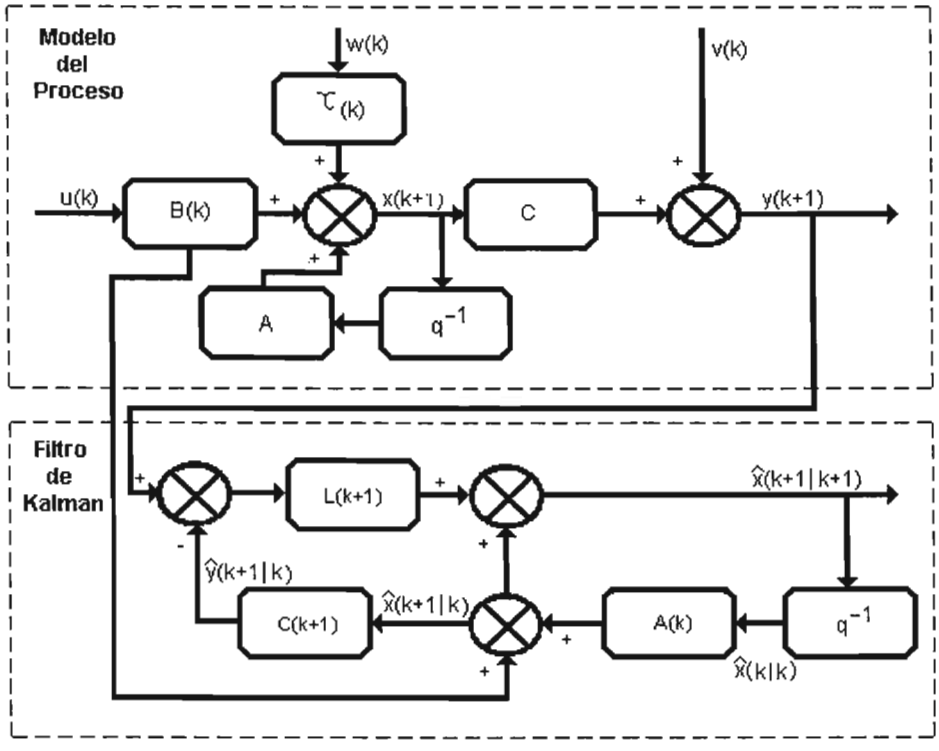


Figura 27 Filtro de Kalman

- Filtro de Kalman

En el *Filtro Kalman* se definen tres modelos, al conjunto de los cuales se le conoce como *modelo del proceso* y dos fases que constituyen el *Filtro de Kalman*.

a) Modelo del sistema. Describe la evolución en el tiempo de la cantidad que se desea estimar, expresada mediante un vector de estado $x(k)$. La transición entre estados $(x(k)) \rightarrow x(k+1)$ se caracteriza por la matriz de transición $A(k)$ y la adición de un ruido gaussiano $w(k)$ de media cero y con matriz de covarianza Q.

$$x(k+1) = A(k)x(k) + B(k)u(k) + \tau(k)w(k), w(k) \rightarrow N(0,Q)$$

b) Modelo del sensor o de medición. Relaciona el vector de medida $y(k)$ con el estado del sistema $x(k)$ a través de la matriz de medición $C(k)$ y la adición de un ruido gaussiano $v(k)$ con una matriz de covariancia R.

$$y(k) = C(k)x(k) + v(k), v(k) \rightarrow N(0,R)$$

c) Modelo a priori. Describe el conocimiento previo sobre el vector de estado en el instante $x(0)$ en cuanto a valor esperado y matriz de covarianza $P(0)$.

$$E[x(0)] = \hat{x}(0), conv[x(0)] = P(0)$$
$$E[v(i)w^T(j)] = 0$$

d) Fase de propagación. En esta fase, se pretende predecir cual va a ser el nuevo valor de la cantidad que se desea estimar. Para ello, la estimación del estado anterior

$$\hat{x}(k+1|k) = A(k)\hat{x}(k|k) + B(k)u(k)$$
$$P(k+1|k) = A(k)P(k|k)A^T(k) + \tau Q \tau^T$$

e) Fase de actualización. En esta fase, se calcula el nuevo vector de estado $\hat{x}(k+1|k+1)$ y su matriz de covarianza $P(k+1|k+1)$. Para ello, se utiliza la covarianza predicha para calcular la ganancia de Kalman $L(k)$, escalando ésta por el valor del residuo de medición $y(k+1) - C(k+1)\hat{x}(k+1|k)$, o sea por la estimación del error cometido en la predicción, y sumándose al valor de estado predicho $\hat{x}(k+1|k)$para calcular el nuevo vector de estado $\hat{x}(k+1|k+1)$.

$$L(k+1) = P(k+1|k)C^T(k+1)\left[C(k+1)P(k+1|k)C^T(k+1) + R\right]^{-1}$$
$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + L(k+1)\left[y(k+1) - C(k+1)\hat{x}(k+1|k)\right]$$
$$P(k+1|k+1) = [1 - L(k+1)C(k+1)]P(k+1|k)$$

El *Filtro Kalman* soluciona el problema de estimar un estado x del sistema descrito por:

$$x_{k+1} = A_k x_k + Bu_k + w_k \qquad (1)$$

A partir de mediciones ruidosas z de la forma:

$$z_k = H_k x_k + v_k \qquad (2)$$

Se asume que w_k y v_k son procesos aleatorios independientes, blancos y con distribución de probabilidad Gaussiana, es decir:

$$\begin{aligned} w_k &\in N(0,Q) \\ v_k &\in N(0,R) \end{aligned} \quad (3)$$

Se designa con $\hat{x}_k^-$ a la estimación a *priori* del vector de estados en el paso k basado en el conocimiento del proceso anterior al paso k, y con $\hat{x}_k$ a la estimación a *posteriori* del estado en el paso k dada la medición z_k. Se definen los errores de ambas estimaciones como:

$$e_k^- = x_k - \hat{x}_k^- \quad (4)$$

$$e_k = x_k - \hat{x}_k \quad (5)$$

Así, las correspondientes matrices de covarianza de errores serán respectivamente:

$$P_k^- = E\left[e_k^- e_k^{-T}\right] \quad (6)$$

$$P_k = E\left[e_k e_k^T\right] \quad (7)$$

Para obtener las ecuaciones correspondientes del filtro Kalman, se parte del objetivo de encontrar una ecuación que compute una estima a *posteriori* del estado $\hat{x}_k$, como una combinación lineal de una estima a priori y una diferencia marcada entre la medición presente z_k y una predicción de la medición $H_k \hat{x}_k^-$, es decir:

$$\hat{x} = \hat{x}_k^- + k\left(z_k - H_k \hat{x}_k^-\right) \qquad (8)$$

A la diferencia se le denomina *innovación* de la medición, o *residuo*. Refleja la discrepancia entre la medición predicha y la medición real.

La matriz de peso k debe ser elegida para minimizar la covarianza del error a *posteriori* P_k. Reemplazando en la ecuación (8) en la expresión del error e_k en (5) y luego en (7), y tomando la derivada respecto a k, e igualando a cero se obtiene:

$$K_k = P_k^- H_k^T \left(H_k P_k^- H_k^T + R_k\right)^{-1}$$
$$K_k = \frac{P_k^- H_k^T}{H_k P_k^- H_k^T + R_k}$$

Con lo anterior se puede observar que cuando la covarianza del ruido de medición R_k tiende a cero, la ganancia k pesa a los residuos más levemente. Específicamente

$$\lim_{R_1 \to 0} K_k = H_k^{-1}$$

Por otra parte, cuando la covarianza P_k^- del error de la estima a priori tiene a cero, la ganancia K pesa a los residuos más elevadamente

$$\lim_{P_1^- \to 0} K_k = 0$$

En otras palabras, a medida que la covarianza del error de medición R_k , se aproxima a cero, se confía más en la medición real z_k y a la vez se confía menos en la predicción $H_K \hat{x}_k^-$ de la medición. Por otra parte, a medida que la covarianza

P_k^- del error de la estima a priori se a próxima a cero, se confía menos en la medición real z_k y cada vez más en la predicción $H_K \hat{x}_k^-$.

Las ecuaciones del *Filtro Kalman* pueden dividirse en dos grupos:

a) *Actualización temporal.* Proyectan hacia delante en el tiempo la estima del estado presente y de la covarianza del error para obtener una estima priori para la próxima iteración.

b) Actualización de las mediciones. Proveen una retroalimentación incorporando una nueva medición en la estima a priori para obtener una estima a posteriori mejorada.

Para estimar los parámetros necesarios del *Filtro Kalman* se considera una estructura de regresor lineal.

$$y(n) = \Phi^T(n)\Theta + e(n) \qquad (9)$$

Escribiéndose como ecuaciones de estado se obtiene:

$$\begin{aligned} x(n+1) &= x(n) \\ y(n) &= \Phi^T(n)x(n) + e(n) \end{aligned} \qquad (10)$$

Donde el vector de estados $x(n)$ en el vector de parámetros Θ. La estima óptima de estados viene dada por el *Filtro Kalman* aplicada a la ecuación (10), y coincide con la estima de mínimos cuadrados recursivos. Una forma de modificar el algoritmo de manera que pueda seguir parámetros que varíen en el tiempo es cambiar la ecuación de estado en (10) por:

$$x(n+1) = x(n) + w(n) \qquad E\left[w(n)w^T(s)\right] = Q\delta_{n,s}$$

La matriz de covarianza Q puede usarse para describir cuán rápido se espera que varíen las componentes de Θ.

Las ecuaciones del filtro Kalman para este caso son:

$$\hat{\Theta}(n) = \hat{\Theta}(n-1) + K(n)\varepsilon(n)$$
$$\varepsilon(n) = y(n) - \Phi^T(n)\hat{\Theta}(n-1)$$
$$K(n) = \frac{P(n)\Phi(n) = P(n-1)\Phi(n)}{1 + \Phi^T(n)P(n-1)\Phi(n)}$$
$$P(n) = \frac{P(n-1) - P(n-1)\Phi(n)\Phi^T(n)P(n-1)}{\left[1 + \Phi^T(n)P(n-1)\Phi(n)\right] + Q}$$

Por último y a manera de resumen se puede definir al *Filtro Kalman* como una técnica o procedimiento, iterativo, usado para estimar los parámetros correctos del modelo de un proceso. Es un algoritmo para optimizar sistemas que varían en el tiempo

IV.V Filtro Kalman Extendido

El *Filtro Kalman Extendido* es una variante del filtro Kalman pero aplicado a procesos no lineales, es decir; a procesos de la forma:

$$x(k+1) = f(x(k)), u(k), w(k)$$
$$z(k+1) = h(x(k+1)), v(k+1)$$

Se efectúa una linealización de las funciones no lineales en torno a la estimación en el instante actual de los estados y los ruidos del modelado:

$$x(k+1) = \hat{x}(k+1 \mid k) + A \cdot \left(x(k) - \hat{x}(k) \right) + W \cdot w(k)$$
$$z(k+1) = \hat{z}(k+1 \mid k) + H \cdot \left(x(k+1) - \hat{x}(k+1 \mid k) \right) + V \cdot v(k+1)$$

Donde:

$$A_k = \frac{\partial f_k}{\partial x_k} | \left(\hat{x}_k, u_k, 0 \right)$$

$$W_k = \frac{\partial f_k}{\partial w_k} | \left(\hat{x}_k, u_k, 0 \right)$$

$$H_k = \frac{\partial h_k}{\partial x_k} | \left(\hat{x}_{k+1|k}, 0 \right)$$

$$A_k = \frac{\partial h_k}{\partial v_k} | \left(\hat{x}_{k+1|k}, 0 \right)$$

Son matrices Jacobianas.

CAPÍTULO V

DESARROLLO DEL SISTEMA DE SEGUIMIENTO DE OBJETOS MÓVILES UTILIZANDO FILTRO DE KALMAN

CAPÍTULO V

En los últimos años, la tecnología ha volcado sus esfuerzos en la optimización de tareas que permitan al ser humano desempeñarse en actividades diversas, confiando a su vez varias de ellas a máquinas capaces de realizarlas en poco tiempo y de forma eficaz.

Los sistemas de vigilancia, tan importantes como indispensables para la protección y seguridad que rodea al ser humano no han escapado al acelerado desarrollo de la tecnología. Por lo cual, se han incorporado diversas técnicas de análisis y optimización que hacen de los sistemas de vigilancia una herramienta básica en el seguridad.

V.I Componentes electrónicos y programas necesarios para el desarrollo del sistema de seguimiento de objetivos móviles usando Filtro Kalman

Para la elaboración del presente proyecto de tesis fueron necesarios diversos elementos que para una mejor descripción, se catalogaron en dos secciones conformadas por componentes electrónicos (hardware) y programas (software).

- Elementos de Hardware

a) Computadora

Como características mínimas en la computadora utilizada para la implantación del sistema de seguimiento se establecen las siguientes:

- Procesador de 2.8 GHz.
- Memoria RAM de 256 Mb.
- Capacidad de Almacenamiento de 20 GB.

b) Framme Grabber (adquisición de datos)

Es indispensable para una de las rutas de adquisición de imagen descritas anteriormente contar con este dispositivo, ya que es por medio de él

se obtiene información relevante para llevar a cabo cálculos, entre otros procesos.

Las características que debe reunir el dispositivo *Frame Grabber*[9] son:

- Alta velocidad de adquisición de píxeles (operación en tiempo real, obtención y despliegue de información).
- Entrada compatible con el formato de salida de la cámara asociada o en su defecto capacidad de adaptación para dicha salida.
- Opciones de señales de entrada/salida para control y sincronización con la cámara.
- Capacidad de acceso a funciones internas del dispositivo para obtención de datos, manipulación de datos o configuración de las mismas.

Por otro lado, el trabajo de investigación realizado para la elaboración del sistema de seguimiento permitió encontrar un camino alterno que evita el uso de la tarjeta Frame Grabber. Este nuevo camino se hace posible gracias a las características de funcionamiento del elemento de software conocido como Active X Imaging Source[10] el cual detecta cualquier cámara de video como dispositivo electrónico conectado a la computadora ya sea mediante el puerto USB o bien mediante la tarjeta Frame Grabber.

c) Cámara de video

Las etapas de operación de una cámara pueden dividirse en etapa de adquisición y etapa de formato de salida de video.

La etapa de adquisición queda definida principalmente por el sensor que adquiere la imagen.

La etapa de formato de salida estará definida con base en la forma de conexión establecida entre la cámara y la computadora, es decir; si la cámara está conectada mediante el dispositivo Frame Grabber o el puerto USB el formato de salida será el que corresponda a las especificaciones de configuración establecidas por el fabricante de cada dispositivo.

[9] *Frame Grabber* . Tarjeta digitalizadora de imagen.

[10] *Active X Imaging Source.* Herramienta utilizada para el despliegue de imagen.

En ambos casos el formato de salida de video para el presente trabajo de tesis estará definido por el estándar RGB 24, para el caso en el que se utilice el la tarjeta Frame Grabber algunos de los formatos de video serán:

- PAL (*Phase Alternate Line*). Formato utilizado para hacer referencia a sistemas y señales compatibles con una técnica de modulación específica
- NTSC (*National Television Systems Committee*). Estándar establecido para televisiones en blanco y negro en 1941 y posteriormente a color en 1953. El Formato NTSC es utilizado para hacer referencia a los sistemas y señales compatibles con la técnica de modulación específica de color.
- Y/C (S-VHS). Interfaz análoga de video en el cual la información del color es llevada de forma separada de la información referente al brillo. Dos partes de un cable son utilizados y denotados como *Y* y *C* o *Y/C*.

Por lo anterior se propone una cámara *Imaging Source* con las características siguientes:

- Sensores CCD
- Capacidad de salida de video en formato compatible con la entrada del siguiente dispositivo.
- Opción de entrada/salida para sincronización con dispositivos externos.

- Elementos de Software

a) Sistema Operativo Windows

De manera coloquial decimos que como *Software* se conoce a la parte intangible de una computadora pero que a su vez es indispensable su estancia en la misma ya que es éste el encargado de establecer un canal de comunicación entre el hardware y el usuario, y es este mismo el que permite que el "ambiente de trabajo" para el usuario sea más o menos *amigable* que otro.

Como características de software se requiere:

- Sistema operativo *Windows 2000 Professional.*
- Software de instalación de *Frame Grabber* compatible con el sistema operativo que posea la computadora (Controladores de reconocimiento del periférico con el sistema operativo)
- Software de instalación de la cámara en caso de utilizarse el puerto USB (Controladores de reconocimiento del periférico con el sistema operativo)

b) Delphi

Delphi es una herramienta de desarrollo rápido de aplicaciones (RAD).

Cuenta con un entorno de desarrollo integrado (IDE) y de una biblioteca de componentes visuales (VCL), cuya implementación basada en el paradigma de la programación orientada a objetos, facilita el desarrollo de aplicaciones bajo sistemas Windows.

La versión de la herramienta de desarrollo Delphi utilizada para desarrollar el sistema de seguimiento es Delphi 7.0.

c) Active X Imaging Source

Herramienta necesaria para desplegar las imágenes arrojadas a través de la cámara en tiempo real.

Para poder utilizarlo simplemente se agrega el componente dentro de los elementos presentados como Active X en el ambiente de desarrollo Delphi y se instala como cualquier otro elemento del tipo Active X que desea ser utilizado y reconocido por el entorno Delphi .

V.II Descripción de elementos electrónicos y programas utilizados para el desarrollo del sistema de seguimiento de objetos móviles utilizando filtro Kalman

- Cámara de Video "Imaging Source"

Una imagen digital es cualquier imagen fija o en movimiento que se captura a través de un medio electrónico y que se representa como un archivo de información leído como una serie de impulsos electrónicos.

EL principio de funcionamiento de cualquier cámara digital se divide básicamente en tres partes:

- Captura de la imagen.

La imagen se obtiene a través de distintas componentes. La parte óptica de la cámara que se encarga de recoger la luz con la mayor precisión y calidad posibles. Las cámaras digitales usan un elemento sensible a la luz que capta las imágenes, las variaciones de luz, colores y texturas traduciéndolas en pulsos eléctricos. El más común es el CCD (Change Coupled Device) o dispositivos adaptadores de carga. El CCD sólo puede captar imágenes en blanco y negro, para producir los colores la imagen debe pasar por tres filtros (rojo, verde y azul).

Todas las cámaras incorporan un Iris para ajustar la exposición y las ganancia eléctrica del CCD.

Antes del CCD se antepone un filtro óptico para eliminar la parte del espectro no visible, lo que elimina el infrarrojo y el ultravioleta.

- Digitalización

Para digitalizar la imagen es necesario pasarla a un formato susceptible de ser comprimido y almacenado en cinta. Se realiza utilizando un convertidos analógico digital que consiste en un chip que toma muestras de la señal a intervalos fijos de tiempo (frecuencias de muestreo), a cada muestra se le asigna un valor dependiendo de su amplitud.

- Procesado de Imagen

Consiste en efectos especiales tales como zoom, estabilización digital, etc.

El uso de una cámara de video digital para el presente trabajo de tesis es básicamente indispensable, por medio de la cámara serán capturadas

cada una de las imágenes que posteriormente se desplegarán el sistema para el respectivo procesamiento y seguimiento del objeto.

- Frame Grabber

La señal generada por la cámara de video que es un flujo de información analógica compuesta por píxeles que forman líneas, las cuales a su vez forman marcos para finalmente transformarse en secuencias completas de video, requiere manipular cada uno de sus componentes si se desean aplicar algún tipo de procesamiento.

Para manipular *computacionalmente* la señal generada por la cámara, se debe de tener acceso a los píxeles que componen la imagen desplegada por la misma. Para realizar lo anterior, se utilizan dispositivos conocidos como *digitalizadores de imagen* tales como el llamado *Frame Grabber* que son capaces de muestrear el flujo de entrada de video permitiendo el acceso a los componentes de la imagen.

El *Frame Grabber* convierte el flujo de entrada en información de tipo binario, almacenándola en registros de memoria cuya dirección puede ser conocida, por lo que no es difícil para el procesamiento digital tener acceso esta información.

Para el desarrollo de este proyecto es indispensable contar con este dispositivo ya que es por medio de él que se obtiene información relevante para llevar a cabo cálculos que derivan de la manipulación de la información obtenida por este medio.

Las características que debe reunir el dispositivo *Frame Grabber* son:

- Alta velocidad de adquisición de pixeles (operación en tiempo real, obtención y despliegue de información).
- Entrada compatible con el formato de salida de la cámara asociada o en su defecto capacidad de adaptación para dicha salida.
- Opciones de señales de entrada/salida para control y sincronización con la cámara.

- Capacidad de acceso a funciones internas del dispositivo para obtención de datos, manipulación de datos o configuración de las mismas

La figura 28 muestra un esquema de una tarjeta digitalizadota de imagen.

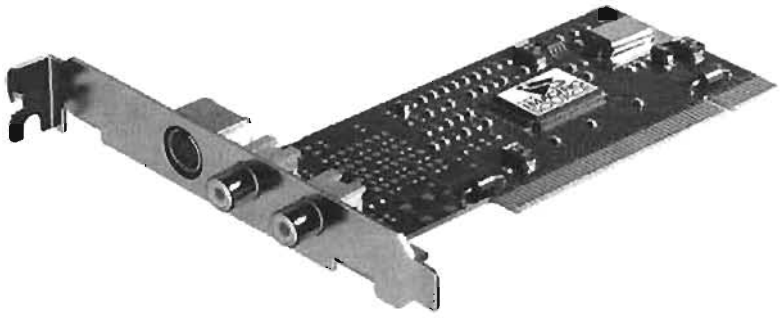


Figura 28 Tarjeta digitalizadora de imagen o Frame Grabber

La tarea del *Frame Grabber* es muestrear y cuantificar la señal de video analógica y almacenar el resultado en la llamada *memoria intermedia del cuadro*.

La frecuencia de muestreo del *Frame Grabber* está dada en el llamado reloj de píxel (píxel clock) y ésta determinará la resolución espacial de la imagen digitalizada y a su vez estará en función del formato de la señal analógica de video de entrada y del formato de la señal digital de vídeo de salida.

Los píxeles digitalizados son almacenados en un buffer de imagen de tipo FIFO, que es capaz de almacenar cuando menos un cuadro digitalizado completo. Este buffer es más utilizado en tanto el bus que comunica al *Frame Grabber* y a la PC tenga un ancho de banda estrecho. No obstante los sistemas de bus modernos como el PCI son tan rápidos que el *Frame Grabber* requiere apenas un buffer de unos cuantos kilobytes para compensar algunas irregularidades en el flujo de datos hacia la computadora a la que está

conectado, de tal forma que el video digitalizado puede ser enviado con prontitud a la memoria de trabajo para procesamiento, o ser dirigido directamente a la tarjeta gráfica para ser visualizado en tiempo real.

- ActiveX Imaging Source

Active X es el nombre que Microsoft ha dado a un grupo de tecnologías y herramientas "estratégicas" orientadas a objetos. Su principal tecnología es el Modelo de Objeto Componente (*Component Object Modelo, COM*).

El objeto que se crea al escribir un programa ejecutable en el entorno Active X es un componente es decir, un programa autosuficiente que puede ejecutarse en cualquier sitio en la red Active X (que es actualmente una red que consta de sistemas tanto Windows como Macintosh). Este componente se conoce como un *Control Active X.*

Dentro del sistema operativo Windows se pueden encontrar diversos archivos con extensión *OCX* (Extensión de Control de enlace e incrustación de Objetos o bien, Object Linking and Embedding Control Extension) es un módulo independiente de programa que puede ser accesado por otros programas en ambiente Windows. Los controles OCX fueron creados para soportar documentos compuestos como lo es el escritorio en un ambiente Windows y son actualmente conocidos como Controles Active X.

Dentro del ambiente de desarrollo del sistema de seguimiento fue necesario la utilización de varios controles active X propios de la herramienta de desarrollo Delphi, sin embargo, fue indispensable agregar un componente más y es el llamado ocx de imaging source.

El ocx de imaging source proporciona al sistema de seguimiento la capacidad de desplegar la imagen captada por la cámara y el Frame Grabber a través de cuadros que permiten a su vez la creación de videos, la toma de fotografías y mayormente importante la tarea de seguimiento a través de los cuadros almacenados en memoria.

La figura 29 muestra el componente ocx de imaging source en el ambiente de desarrollo visual Delphi así como sus propiedades de configuración para su programación y utilización de las aplicaciones de visión.

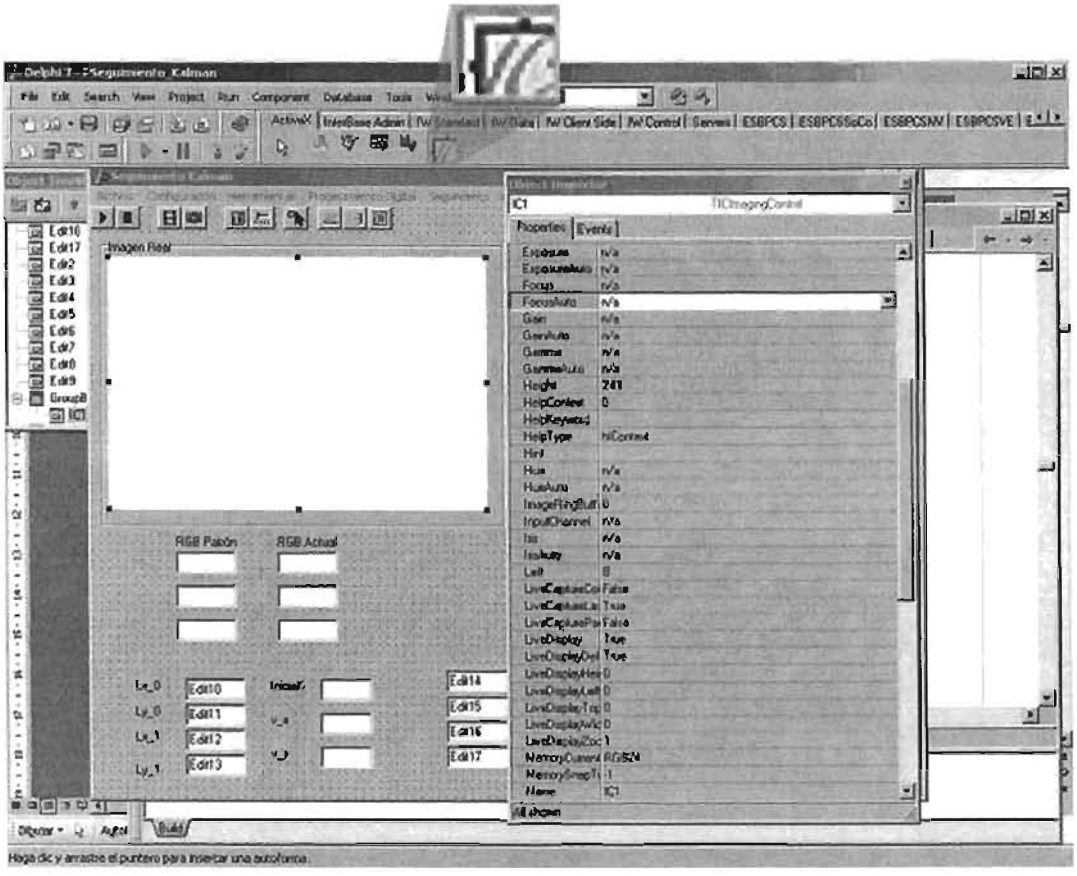


Figura 29 OCX de Imaging Source.

V.III Descripción del proceso de análisis y obtención de resultados del sistema de seguimiento de objetos móviles utilizando filtro Kalman

El patrón por el cual se regirá el proceso de análisis de la imagen una vez capturada mediante una de las rutas descritas anteriormente, quedará establecido de forma general en el diagrama 30.

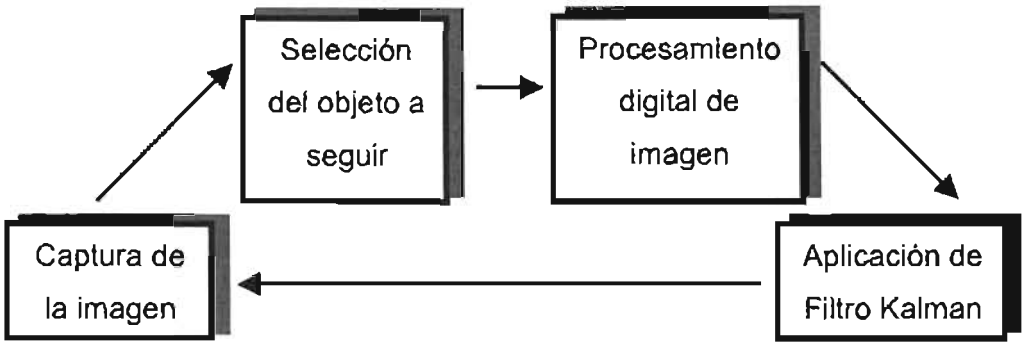


Figura 30 Etapas del proceso de análisis y obtención de resultado del sistema de seguimiento

V.IV Captura de imagen del sistema de seguimiento de objetos móviles utilizando filtro Kalman

Para describir en su totalidad el sistema de seguimiento implementado, primeramente se deben de entender las fases por las cuales la imagen es captada.

El diagrama 31 describe las etapas principales del proceso de captura de la imagen.

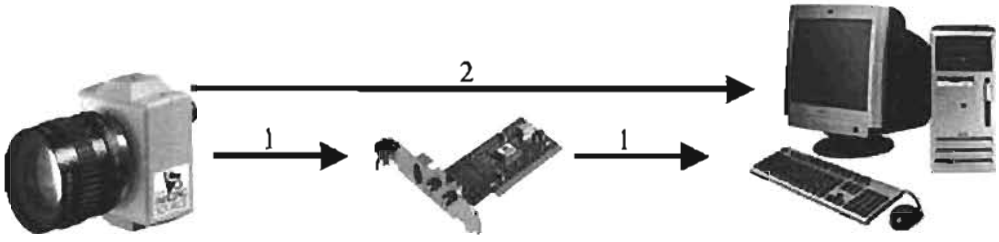


Figura 31 Diagrama de sistema de seguimiento

La ruta marcada con el número 1 muestra una imagen que es capturada por una cámara para posteriormente ser transmitida a la tarjeta Frame Grabber la cual por medio de cuadros (*frames*) enviará la información a la computadora para que ésta a su vez realice el procesamiento y análisis respectivos.

La ruta marcada con el número 2 muestra una imagen que al ser capturada es inmediatamente enviada a la computadora para que sean realizadas las tareas de procesamiento y análisis.

Cualquiera de las dos rutas descritas anteriormente puede ser utilizada dentro por el sistema de seguimiento, aunque contrario a lo que pareciese, la ruta descrita con el número 1 es mayormente recomendada debido a las características de manejo de imagen que presenta la tarjeta Frame Grabber.

V.V Proceso de selección del objeto a seguir del sistema de seguimiento de objetos móviles utilizando filtro Kalman

El ambiente sobre el cual el sistema de seguimiento puede trabajar de forma casi inequívoca es un ambiente en el que se pueden distinguir claramente los objetos que lo componen.

El sistema de seguimiento tiene la facultad de elegir de la imagen mostrada por la cámara un objeto en particular. Con ayuda del *Mouse* de la computadora se puede iniciar el proceso de selección.

Al momento de posicionarse en un punto de la imagen desplegada y arrastrando dicho periférico se formará una figura geométrica de cuatro vértices que simularán un cuadrado o rectángulo dejando en su interior al objeto deseado estableciendo así la llamada *región de interés*.

De esta forma, se delimitará la zona de interés sobre la imagen original evitando trabajar con la imagen completa en procesos posteriores.

Como se mencionó anteriormente, el formato de salida de video a utilizar es el estándar RGB 24 por lo que cada pixel está conformado por tres bytes de profundidad lo que significa que cada pixel contendrá un componente en B, un componente en G y un componente en R de forma consecutiva tal como se muestra en la figura 32.

Figura 32 Representación del formato estándar RGB 24 en un pixel

Para obtener la intensidad de niveles RGB en la región de interés, se requiere encontrar el número total de pixeles contenidos en dicha región así como el promedio de valores RGB. El dividir el promedio de valores RGB entre el número total de pixeles dará como resultado el valor RGB representativo de dicha región.

El número total de pixeles se obtiene multiplicando el ancho por el alto de la región de interés.

El promedio de valores RGB se obtiene mediante la suma algebraica de los niveles RGB de los pixeles que conforman la región de interés.

La figura 33 muestra la selección de un objeto dentro de una imagen desplegada.

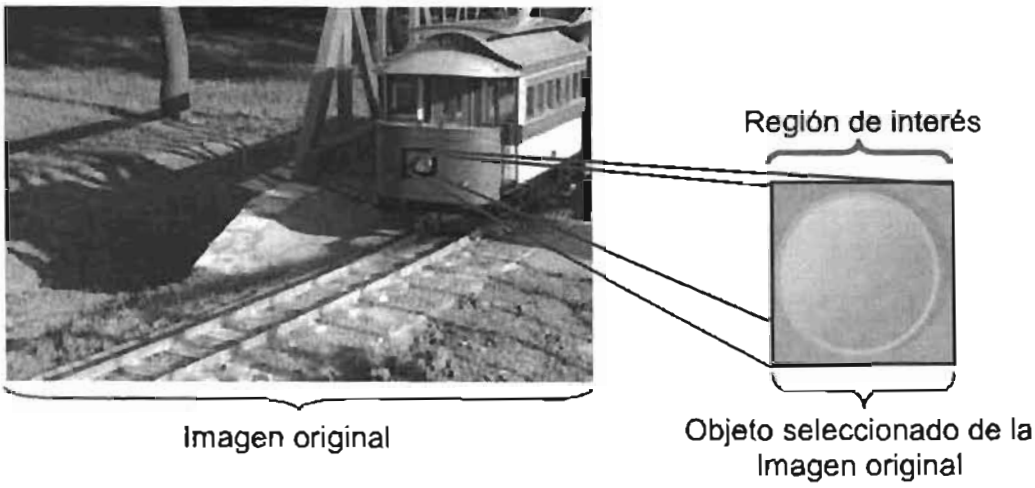


Figura 33 Selección del objeto de interés

La figura 34 muestra la forma en la que es obtenido el promedio de la representación RGB de la región de interés.

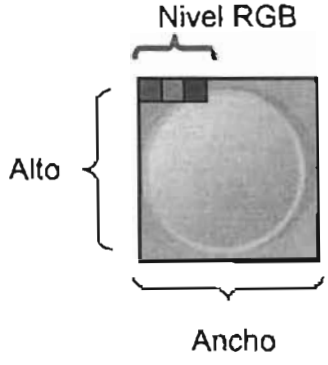


Figura 34 Intensidad de niveles RGB representativo

La representación digital de la imagen por medio de píxeles, permite asignar una posición a cada píxel en el área de despliegue de la imagen.

La obtención de las posiciones representativas (x, y) están referidas al centroide gráfico de la región de interés. La posición x del centroide se obtiene dividiendo el ancho de la región de interés entre 2, la posición y es el resultado de la división del alto entre 2.

Estas coordenadas se encuentran al momento en el que es soltado el Mouse dando como resultados las coordenadas iniciales (x_0, y_0) del objeto tal y como se muestra en la figura 35.

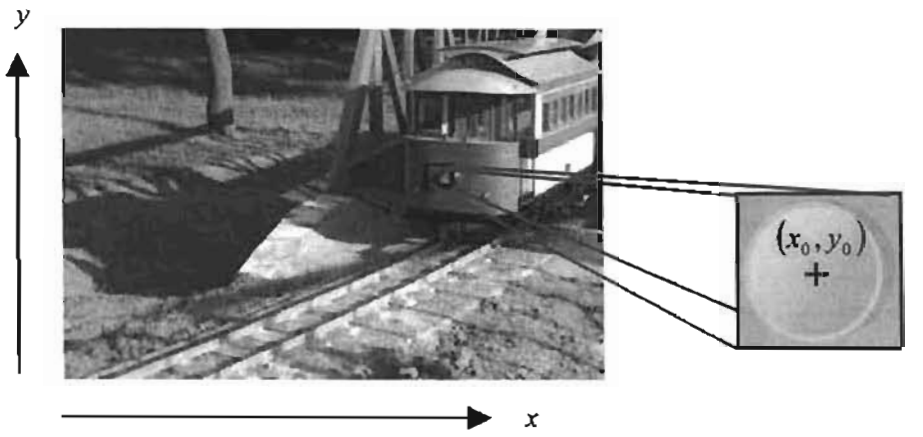


Figura 35 Obtención de la posición representativa

V.VI Procesamiento digital de imagen del sistema de seguimiento de objetos móviles utilizando filtro Kalman

Una vez que se ha realizado el proceso de selección del objeto, el sistema de seguimiento podrá realizar diferentes tipos de procesamiento sobre la imagen delimitada y particularmente, sobre la región sobre la cual se encuentra la máscara de búsqueda.

Un tipo de procesamiento será el realizado a través de la aplicación del filtro *Sobel* el cual podrá ser observado seleccionando el botón *"Sobel"*. De este puede observarse también el histograma que representa los diferentes niveles de grises de la imagen. Este histograma se obtiene presionando el botón *"Histograma"* ubicado en el menú principal, cabe señalar que se puede obtener una vez aplicado el filtro *Sobel* o bien cuando la imagen no ha sufrido ningún tipo de procesamiento o selección de zona.

El tipo de procesamiento que se lleva a cabo una vez que se ha presionado el botón de *"Seguimiento"* difiere al anterior en lo que se refiere a la obtención de niveles de gris. El obtener dichos valores permitirán que la etapa siguiente obtenga las condiciones iniciales necesarias para poder comenzar con el proceso de cálculo.

V.VII Filtro Kalman del sistema de seguimiento de objetos móviles utilizando filtro Kalman

Tal y como se explicó en el capítulo IV apartado IV, el filtro de Kalman es un filtro predictivo de tal forma que para poder hacer uso de él, se requieren condiciones iniciales de velocidad y posición.

Para encontrar la posición inicial simplemente se requiere del uso del Mouse en el proceso de selección del objeto tal y como se explicó anteriormente, al dar click sobre la imagen las coordenadas (x, y) son encontradas convirtiéndose así en las coordenadas (x_0, y_0) para el filtro de Kalman.

En el caso de la velocidad, una vez que se obtiene la posición del objeto está puede ser calculada utilizando la ecuación:

$$v = \frac{p}{t}$$

Para poder encontrar la velocidad es necesario encontrar la siguiente posición del objeto, ésta se encuentra cuando es desplegado el siguiente *frame* en pantalla utilizando la máscara de búsqueda se hace nuevamente el promedio de niveles de gris establecido un umbral. Así es encontrando un

objeto con características similares al seleccionado lo que establece las coordenadas (x_1, y_1) que marcan la siguiente posición del objeto p.

Con p_0 y p_1 se encuentra la velocidad a la que se desplaza el objeto a seguir obteniendo de esta forma las condiciones iniciales del filtro de Kalman.

El proceso anterior se realiza de forma cíclica cada vez que el filtro Kalman es llamado y no deja de realizarse hasta el filtro es detenido.

Para una mejor y mayor comprensión del funcionamiento e implementación del filtro Kalman se utilizará el siguiente ejemplo.

Estableciendo el modelo del sistema físico por un vector de estados S y un conjunto de ecuaciones llamado el *modelo del sistema*.

El tiempo de observación de cada muestra tiene la forma:

$$t(k) = t_0 + k\Delta T, k = 0,1,...$$

De donde ΔT es el intervalo de muestreo y $S(k)$ el estado $S(tk)$.

Por razones de simplicidad se supone ΔT insignificante por lo que se puede usar un modelo lineal del sistema.

Estado :

$$S(k) = \begin{bmatrix} x(k) \\ y(k) \\ v_x(k) \\ v_y(k) \end{bmatrix}$$

Donde:

$x(k)$ y $y(k)$ Representan la posición en el instante k

$v_x(k)$ y $v_y(k)$ Representan la velocidad en el instante k

Las ecuaciones del sistema estarán dada por:

$$
\begin{aligned}
x(k) &= x(k-1) + v_x(k-1) \\
y(k) &= y(k-1) + v_y(k-1) \\
v_x(k) &= v_x(k-1) \\
v_y(k) &= v_y(k-1)
\end{aligned}
$$

En un instante k la posición es igual a la posición en el instante $k-1$ más un desplazamiento (velocidad por tiempo, con $t=1$).

El modelo del sistema estará definido como:

$$
\begin{aligned}
S(k) &= \Phi(k-1)S(k-1) + u(k-1) \\
\begin{bmatrix} x(k) \\ y(k) \\ v_x(k) \\ v_y(k) \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x(k-1) \\ y(k-1) \\ v_x(k-1) \\ v_y(k-1) \end{bmatrix} + \begin{bmatrix} u_1(k-1) \\ u_2(k-1) \\ u_3(k-1) \\ u_4(k-1) \end{bmatrix}
\end{aligned}
$$

El subíndice $k-1$ en ϕ indica que la matriz de transición ϕ puede ser función del tiempo. En este caso se considera una constante.

$u(k-1)$ es un vector aleatorio Gaussiano con ruido blanco de media cero y matriz de covarianza *Q(k)* y modela el ruido aditivo.

$$
Q(k) = \begin{bmatrix} \sigma_Q^2 & 0 & 0 & 0 \\ 0 & \sigma_Q^2 & 0 & 0 \\ 0 & 0 & \sigma_Q^2 & 0 \\ 0 & 0 & 0 & \sigma_Q^2 \end{bmatrix}
$$

donde σ_Q^2 es un parámetro.

El siguiente elemento en la teoría de la estimación es el modelo de la medida. En cada instante $t(k)$ se tiene una observación ruidosa del vector de estados.

La medida estará dada por:

$$z(k)=\begin{bmatrix} z_x(k) \\ z_y(k) \end{bmatrix}$$

De donde:

$$z_x(k)=x(k)$$
$$z_y(k)=y(k)$$

El modelo de mediciones estará definido como:

$$z(k)=M(k)S(k)+w(k)$$
$$\begin{bmatrix} z_x(k) \\ z_y(k) \end{bmatrix}=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}\begin{bmatrix} x(k) \\ y(k) \\ v_x(k) \\ v_y(k) \end{bmatrix}+\begin{bmatrix} w_1(k) \\ w_2(k) \end{bmatrix}$$

La matriz $M(k)$ controla el cambio de la estimación y puede ser una función dependiente del tiempo. Para este desarrollo se considerará constante.

$w(k)$ es un vector aleatorio Gaussiano de ruido blanco con media cero y matriz de covarianza $R(x)$ y modela el ruido aditivo.

$$R(k)=\begin{bmatrix} \sigma_R^2 & 0 \\ 0 & \sigma_R^2 \end{bmatrix}$$ donde σ_R^2 es un parámetro.

Para establecer las condiciones iniciales del análisis de un objeto en movimiento se ha seleccionado uno cuya posición inicial en el primer cuadro (frame) es:

$$P_0[x_0, y_0]=[150,100]$$

En el segundo cuadro su posición estará definida como:

$$P_1[x_1, y_1] = [155,110]$$

Las condiciones iniciales del objeto, establecidas anteriormente se muestran en la figura 36

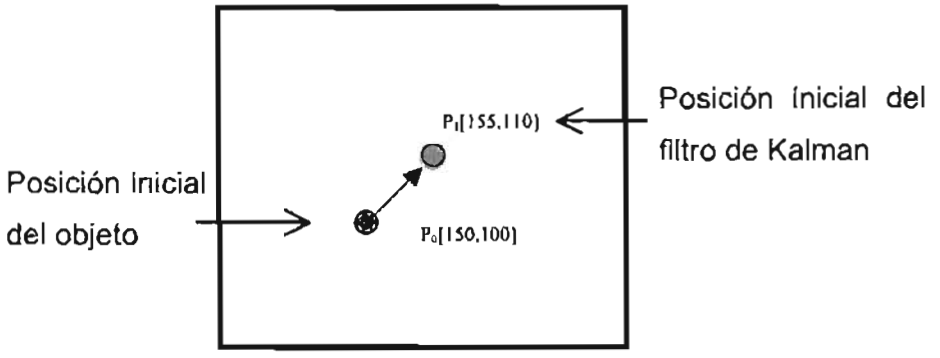


Figura 36 Establecimiento condiciones iniciales

Dadas las posiciones anteriores se pueden calcular las velocidades en los ejes x y y del pixel que se está analizando:

$$v_x = v_{x1} - v_{x0}$$
$$v_x = 5 pix / cuadro$$
$$v_y = v_{y1} - v_{y0}$$
$$v_y = 10 pix / cuadro$$

Se desea calcular mediante el uso del filtro Kalman, la posible ubicación del objeto en el tercer cuadro (frame). El objetivo es obtener un estimado del vector de estados, llamado $\hat{S}$. La parte central del filtro Kalman es el cálculo de la matriz de covarianzas del error al estimar **S** llamada matriz P. El filtro Kalman comienza con un valor inicial de **S** el cual es llamado el valor a priori y se denota $\hat{S}(k-1)$. Su matriz de error de covarianza es $P(k-1)$.

Las condiciones iniciales estarán establecidas como:

$$M(k)=\begin{bmatrix}1&0&0&0\\0&1&0&0\end{bmatrix} \qquad \Phi(k-1)=\begin{bmatrix}1&0&1&0\\0&1&0&1\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

$$\sigma_Q^2 = 0.0001 \qquad Q(k-1)=\begin{bmatrix}0.0001&0&0&0\\0&0.0001&0&0\\0&0&0.0001&0\\0&0&0&0.0001\end{bmatrix}$$

$$\sigma_R^2 = 0.01 \qquad R(k)=\begin{bmatrix}0.01&0\\0&0.01\end{bmatrix} \qquad P(k-1)=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

Se propone que las velocidades en x y y sean constantes, por lo que:

$$v^x(k)=v_{x0}=5\,pix/cuadro$$
$$v_y(k)=v_{y0}=10\,pix/cuadro$$

Se desea estimar $S(k)$, para lo cual Kalman propone extrapolar el estimado de $S(k-1)$ usando la ecuación:

$$\bar{S}(k)=\Phi(k-1)\hat{S}(k-1).....(1)$$

$$\bar{S}(k)=\begin{bmatrix}1&0&1&0\\0&1&0&1\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}155\\110\\5\\10\end{bmatrix} \qquad \bar{S}(k)=\begin{bmatrix}160\\120\\5\\10\end{bmatrix}$$

$$\bar{P}(k)=\phi(k-1)P(k-1)\phi(k-1)'+Q(k-1) \quad .....(2)$$

Sustituyendo valores en la ecuación (2)

$$\bar{P}(k) = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix}$$

$$\bar{P}(k) = \begin{bmatrix} 2.0001 & 0 & 1 & 0 \\ 0 & 2.0001 & 0 & 1 \\ 1 & 0 & 1.0001 & 0 \\ 0 & 1 & 0 & 1.0001 \end{bmatrix}$$

Estableciendo ahora la ecuación

$$K(k) = \bar{P}(k)M(k)'[M(k)\bar{P}(k)M(k)' + R(k)]^{-1} \quad \text{.....(4)}$$

Sustituyendo valores en la ecuación (4)

$$K(k) = \begin{bmatrix} 2.0001 & 0 & 1 & 0 \\ 0 & 2.0001 & 0 & 1 \\ 1 & 0 & 1.0001 & 0 \\ 0 & 1 & 0 & 1.0001 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} *$$

$$\left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 2.0001 & 0 & 1 & 0 \\ 0 & 2.0001 & 0 & 1 \\ 1 & 0 & 1.0001 & 0 \\ 0 & 1 & 0 & 1.0001 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix} \right)^{-1}$$

$$K(k) = \begin{bmatrix} 0.995 & 0 \\ 0 & 0.995 \\ 0.4975 & 0 \\ 0 & 0.4975 \end{bmatrix}$$

Antes de ejecutar la tercera ecuación del filtro kalman se deben obtener las medidas que nos da el sensor, para este ejemplo se realiza un estimado de lo que sería el sensor que toma las medidas, dicho estimado se calcula de la siguiente forma:

$$z_x(k) = x_1 + v_{x1} + ruido\_x$$
$$z_y(k) = y_1 + v_{y1} + ruido\_y$$

Se propone *ruido_x=2* y *ruido_y = -1*

$$z_x(k) = 155 + 5 + 2 = 162$$
$$z_y(k) = 110 + 10 - 1 = 119$$

$$z(k) = \begin{bmatrix} 162 \\ 119 \end{bmatrix}$$

Estableciendo la ecuación

$$\hat{S}(k) = \bar{S}(k-1) + K(k)[z(k) - M(k)\bar{S}(k-1)] \quad .....(3)$$

Sustituyendo valores en la ecuación (3)

$$\hat{S}(k) = \begin{bmatrix} 160 \\ 120 \\ 5 \\ 10 \end{bmatrix} + \begin{bmatrix} 0.995 & 0 \\ 0 & 0.995 \\ 0.495 & 0 \\ 0 & 0.495 \end{bmatrix} \left(\begin{bmatrix} 162 \\ 119 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 160 \\ 120 \\ 5 \\ 10 \end{bmatrix} \right)$$

$$S(k) = \begin{bmatrix} 161.9901 \\ 119.0050 \\ 5.9950 \\ 9.5025 \end{bmatrix} = \begin{bmatrix} 162 \\ 119 \\ 6 \\ 10 \end{bmatrix}$$

Lo que indica que la nueva posición del píxel será:

$$P_2[x2, y2] = [162,119]$$

Estableciendo la ecuación

$$P(k) = [I - K(k)M(k)]\bar{P}(k)[I - K(k)M(k)]' + K(k)R(k)K(k)' \quad \text{.....(5)}$$

Sustituyendo valores en la ecuación (4)

$$P(k) = \left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0.995 & 0 \\ 0 & 0.995 \\ 0.495 & 0 \\ 0 & 0.495 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}\right) \begin{bmatrix} 2.01 & 0 & 1 & 0 \\ 0 & 2.01 & 0 & 1 \\ 1 & 0 & 1.01 & 0 \\ 0 & 1 & 0 & 1.01 \end{bmatrix}$$

$$\left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0.995 & 0 \\ 0 & 0.995 \\ 0.495 & 0 \\ 0 & 0.495 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}\right) + \begin{bmatrix} 0.995 & 0 \\ 0 & 0.995 \\ 0.495 & 0 \\ 0 & 0.495 \end{bmatrix} \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix} \begin{bmatrix} 0.995 & 0 & 0.4975 & 0 \\ 0 & 0.995 & 0 & 0.4975 \end{bmatrix}$$

$$P(k) = \begin{bmatrix} 0.0100 & 0 & 0.0050 & 0 \\ 0 & 0.0100 & 0 & 0.0050 \\ 0.0050 & 0 & 0.5150 & 0 \\ 0 & 0.0050 & 0 & 0.5150 \end{bmatrix}$$

El resultado obtenido en la última ecuación nos indica que la nueva predicción del píxel está dado por $P_2[x2, y2] = [162,119]$, mientras que el resultado obtenido en la ecuación 5 representa la nueva matriz de covarianzas.

Estos resultados se usarán en la siguiente iteración para la ecuación 1, quedando de la siguiente forma:

Ecuación 1 $\bar{S}(k) = \Phi(k-1)\hat{S}(k-1)$

$$\bar{S}(k) = \begin{bmatrix} 0.01 & 0 & 0.005 & 0 \\ 0 & 0.01 & 0 & 0.005 \\ 0.005 & 0 & 0.5026 & 0 \\ 0 & 0.005 & 0 & 0.5026 \end{bmatrix} \begin{bmatrix} 162 \\ 119 \\ 6 \\ 10 \end{bmatrix}$$

La figura 37 muestra la posición final del filtro de Kalman una vez llevado a cabo el proceso de búsqueda.

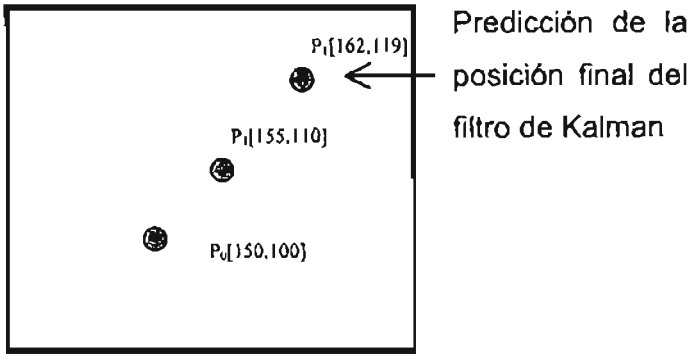


Figura 37 Predicción de la posición final del filtro de Kalman

V.VIII Descripción funcional del sistema de seguimiento de objetos móviles utilizando filtro Kalman (interfaz de usuario)

- Pantalla Principal

  Dividida en cinco zonas principales:

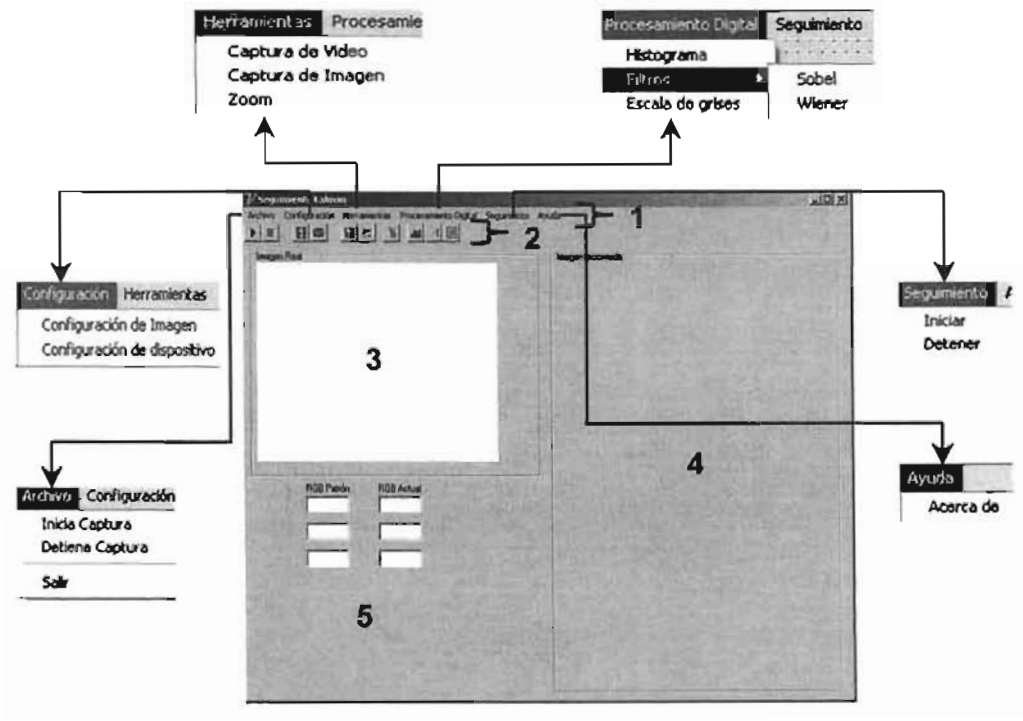


Figura 38 Pantalla principal de interfaz de usuario

1) Zona de menú principal
2) Zona de acceso por botones
3) Zona de despliegue de imagen
4) Zona de procesamiento de imagen
5) Zona de valores RGB

V.IX Descripción de zonas del sistema de seguimiento de objetos móviles utilizando filtro Kalman

- Archivo
  - Despliega la opción *Inicia Captura* la cual permite una vez seleccionado el dispositivo mostrar la secuencia de imágenes capturadas a través de la cámara. De forma simllar, el icono mostrado en la parte inferior izquierda del menú realiza la tarea anteriormente descrita facilitando el manejo de la interfaz hacia el usuario.
  - Despliega la opción *Detiene Captura*. Es habilitada una vez que se ha elegido el dispositivo de trabajo siendo su función principal detener la secuencia de imágenes capturadas a través de la cámara. El segundo icono mostrado en la parte inferior izquierda del menú también es habilitado para la realización de la tarea anteriormente descrita.
  - Despliega la opción *Salir*. Cierra el sistema de seguimiento deteniendo a su vez todas las tareas que se estén llevando a cabo en ese momento.

La figura 39 muestra el elemento Archivo del menú principal e iconos asociados.

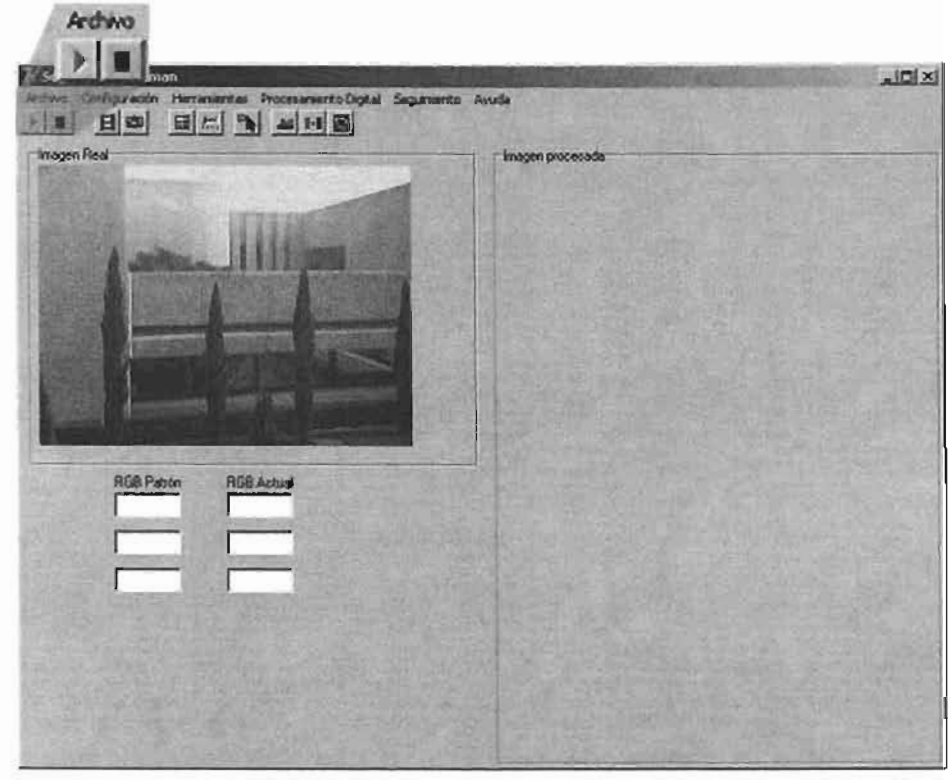


Figura 39 Elemento Archivo del menú e iconos relacionados

- Configuración
  - Despliega la opción *Configuración de Imagen*. Muestra una segunda pantalla que permite una vez mostrada la imagen configurar sus propiedades tales como brillo, contraste, matiz, saturación, etc. que permiten que la imagen mostrada sea adaptada a las condiciones naturales del lugar. El sexto icono en la parte inferior izquierda del menú cuenta con la misma funcionalidad.

La figura 40 muestra la pantalla de configuración de imagen.

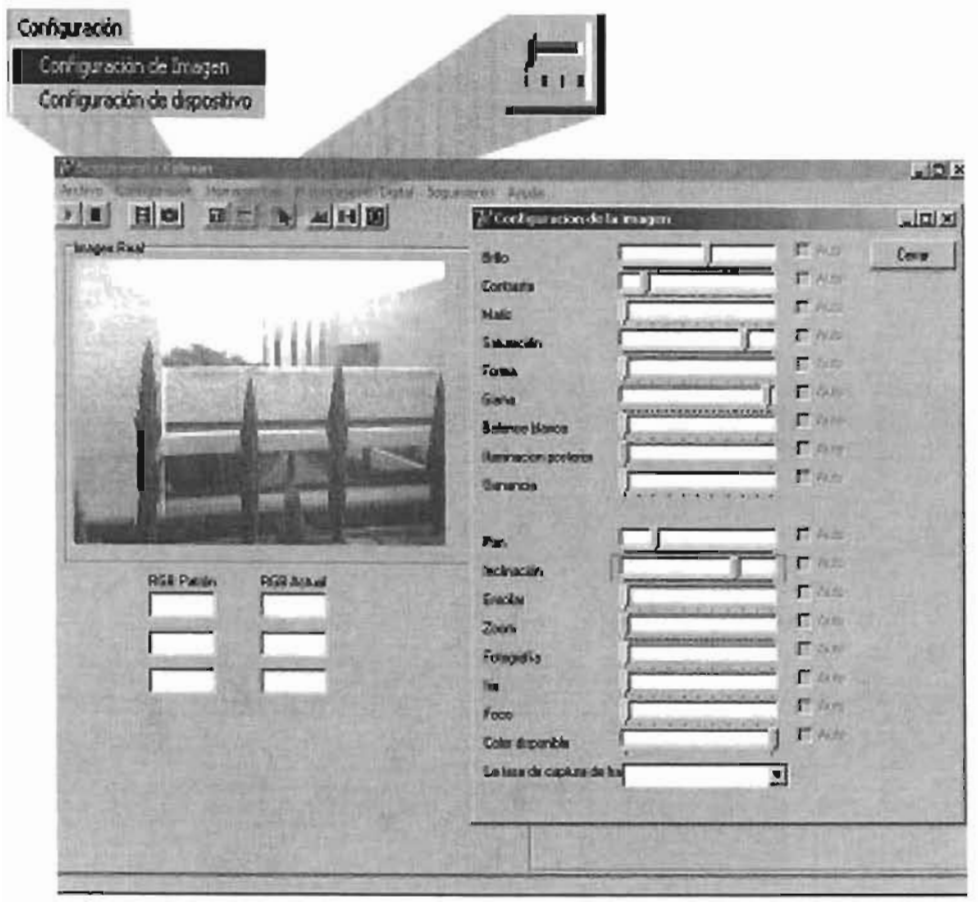


Figura 40 Elemento configuración de Imagen, pantalla de configuración de imagen e icono asociado.

- Despliega la opción *Configuración de Dispositivo.* Muestra una segunda pantalla que permite elegir el dispositivo a utilizar, la norma de video que se habilita cuando el dispositivo seleccionado está conectado a la computadora mediante la tarjeta Frame Grabber y que indica el número de cuadros que podrán ser capturados, el formato de video que representa el número de

píxeles que serán considerados para el despliegue de la imagen y el canal de entrada. El icono asociado a la configuración del dispositivo es el quinto en la parte inferior izquierda del menú.
La figura 41 muestra la pantalla de configuración de dispositivo desplegada y su respectivo icono asociado.

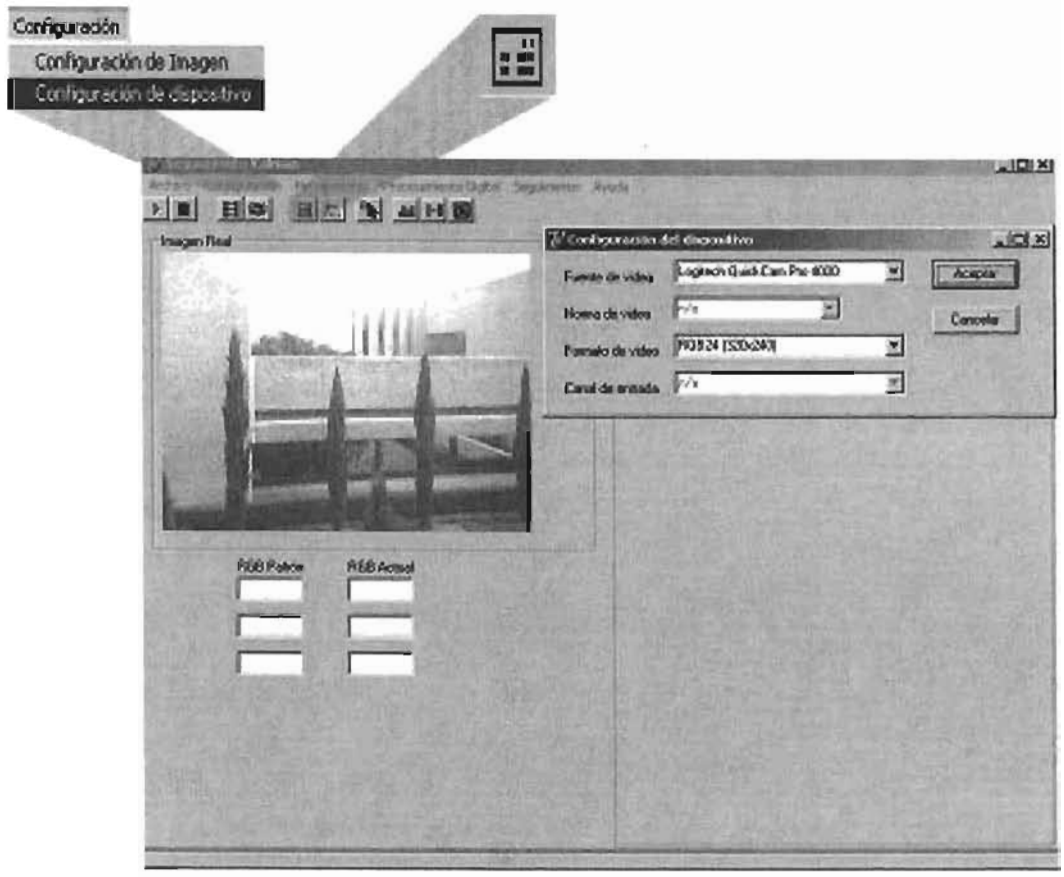


Figura 41 Elemento configuración de dispositivo, pantalla de configuración de dispositivo e icono asociado.

- Herramientas
  - Despliega la opción *Captura de Video*. Muestra una segunda pantalla que permite indicar el codec de video a utilizar así como la ruta en la que se desea almacenar el video con extensión . AVI.

Posee un icono asociado que es mostrado junto con el elemento del menú principal y la pantalla desplegada en la figura 42.

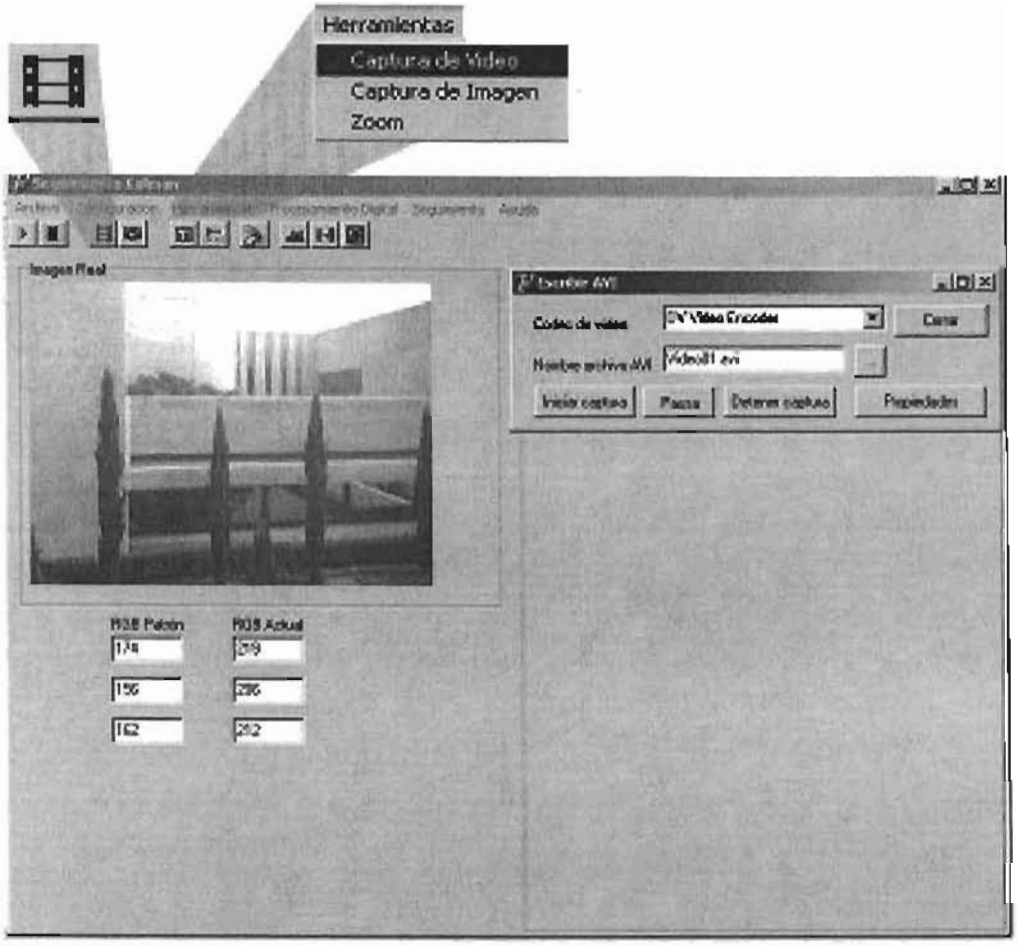


Figura 42 Elemento captura de video, icono asociado y pantalla de almacenamiento de video

A su vez, si se desea cambiar la ruta de almacenamiento del vìdeo, se puede elegir directamente del explorador mostrado los distintos directorios existentes en el disco ello presionando el botón de búsqueda tal y como se muestra en la figura 43.

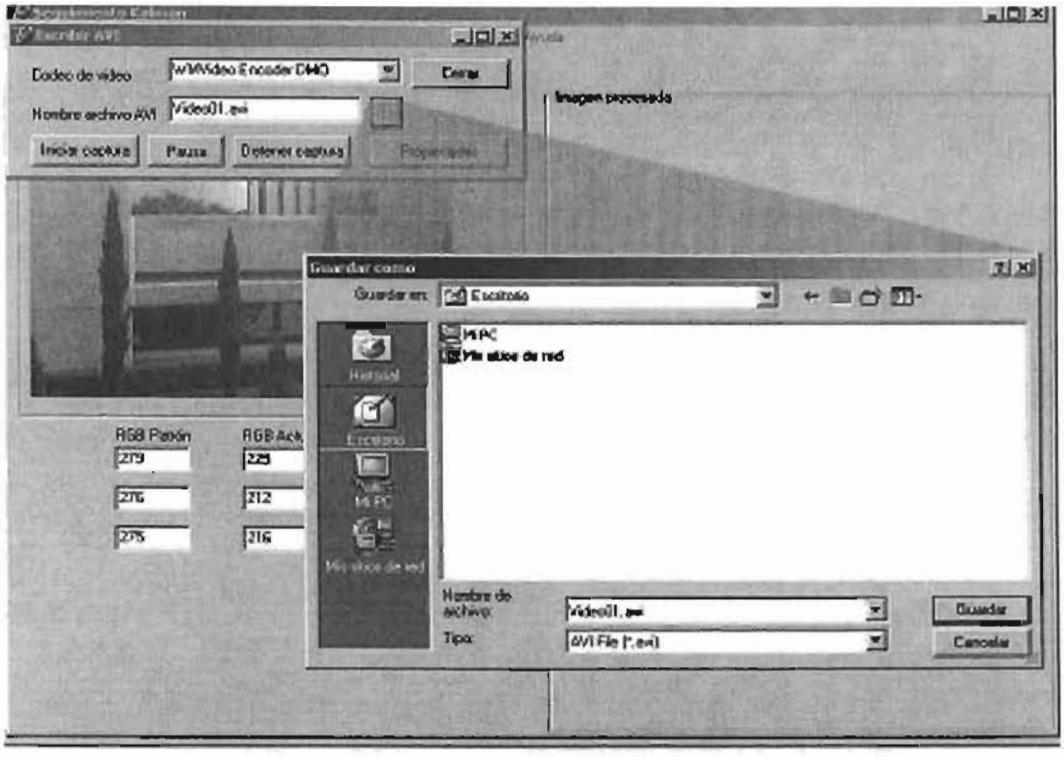


Figura 43 Icono de búsqueda de directorios y pantalla de exploración de ruta.

- Despliega la opción *Captura de Imagen.* Muestra una segunda pantalla que permite elegir la ruta de almacenamiento de la imagen desplegada con la extensión .bmp.

El elemento de captura de imagen, el icono asociado y la pantalla de almacenamiento de imagen se muestran en la figura 44

Figura 44 Elemento captura de imagen, Icono asociado y pantalla de almacenamiento de imagen.

- Despliega la opción Zoom. Tiene la tarea de *acercar* o *alejar* la imagen mostrada en la zona de despliegue para una mejor apreciación.

  El elemento de zoom del menú de herramientas se muestra en la figura 45.

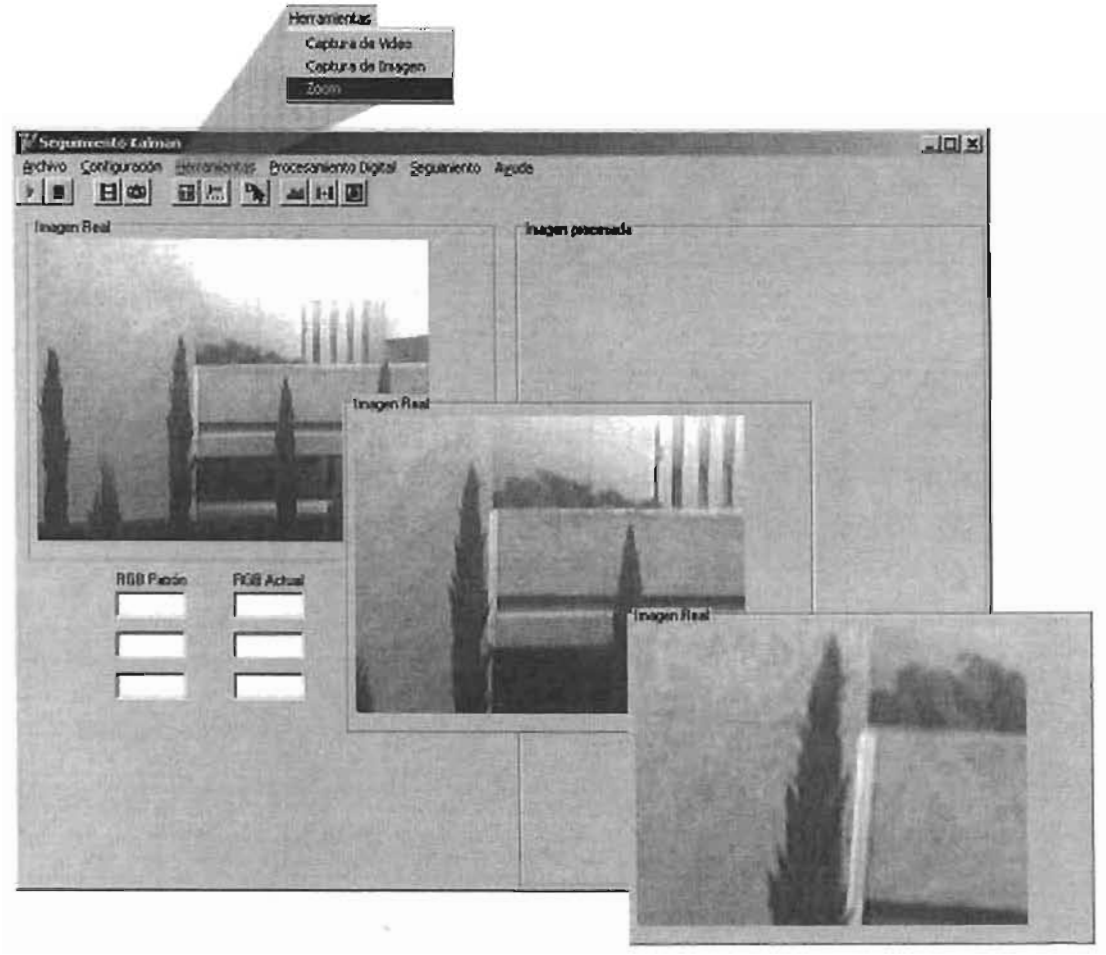


Figura 45 Elemento zoom.

- Procesamiento Digital
  - Despliega la opción *Histograma*. Muestra en la zona de procesamiento de imagen el histograma de los niveles RGB representados por la imagen de interés desplegada en la misma zona.

La figura 46 muestra el elemento Histograma así como el icono asociado.

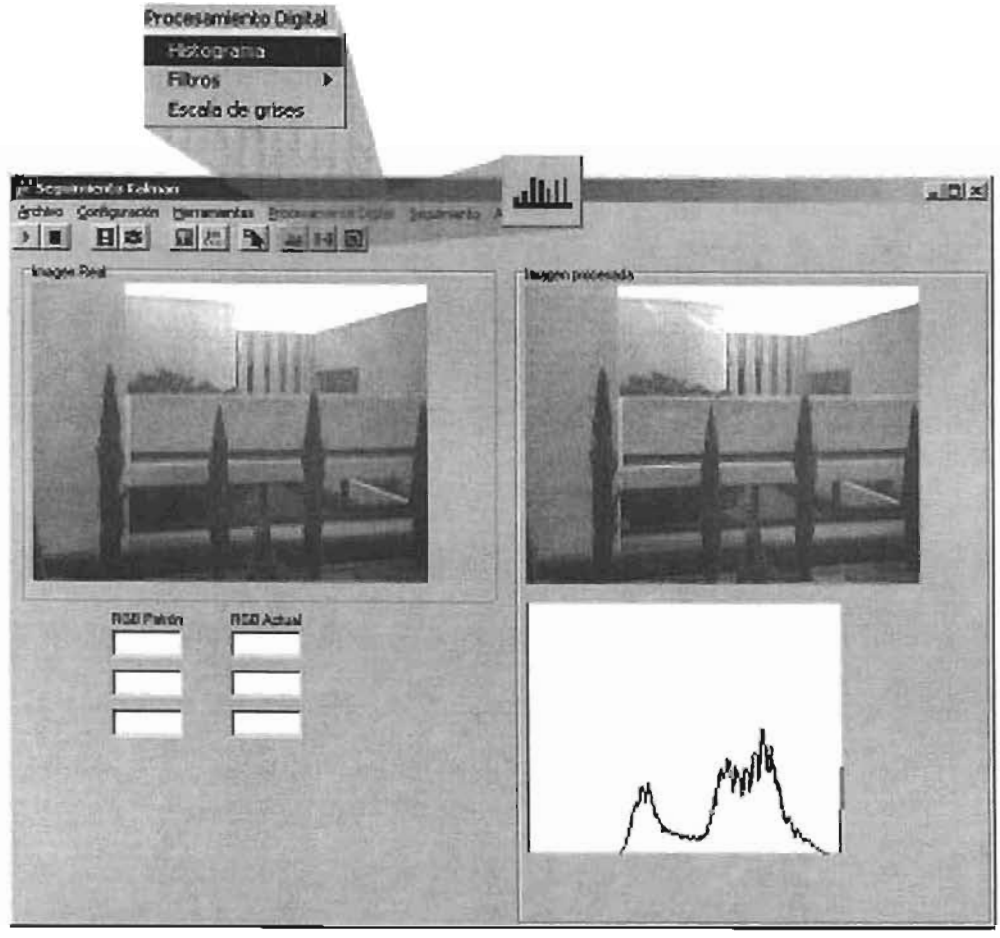


Figura 46 Elemento histograma e icono asociado.

- Despliega la opción *Filtros*. Despliega un submenú de dos opciones 1)Sobel, 2)Wiener. Implementa sobre la imagen desplegada en la zona de procesamiento el filtro seleccionado.

La figura 47 muestra el elemento Filtro e icono asociado.

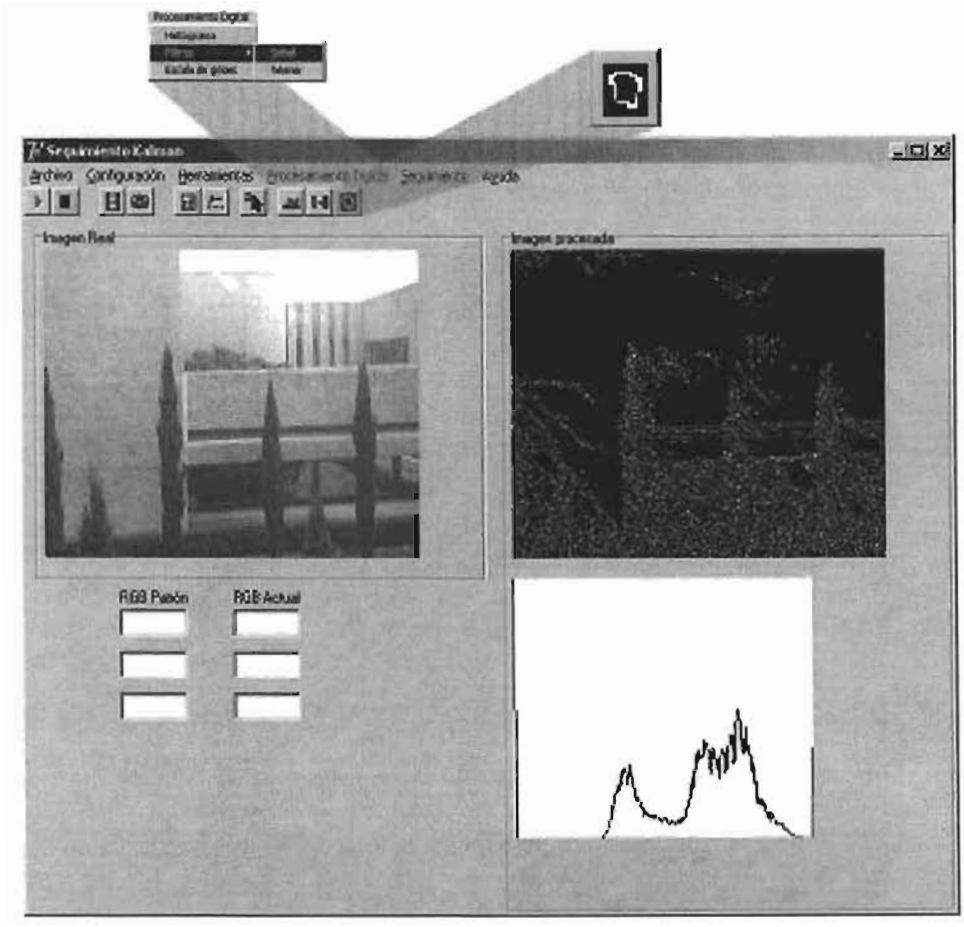


Figura 47 Elemento filtro e icono asociado.

- Despliega la opción *Escala de Grises*. La imagen desplegada en la zona de procesamiento es representada por pixeles en niveles de gris, es decir; la imagen es cambiada de formato RGB a niveles de gris.

La figura 48 muestra el elemento Escala de Grises e icono asociado.

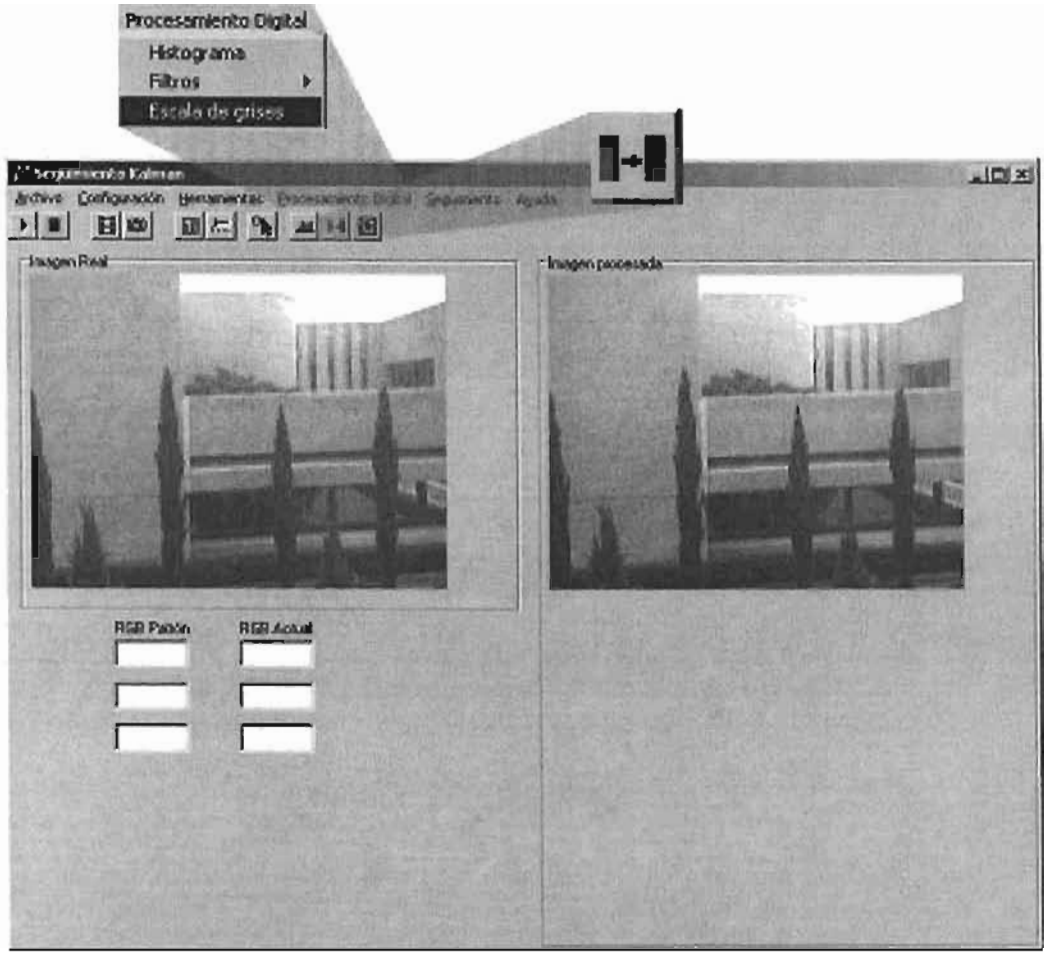


Figura 48 Elemento escala de grises e icono asociado.

- Seguimiento
  - Despliega la opción Iniciar. Una vez generada el área de seguimiento sobre la zona de despliegue de imagen comienza el proceso de seguimiento mediante filtro Kalman y procesamiento

de imagen, mostrando a su vez mediante un cuadro el resultado obtenido y en la zona de valores RGB los niveles RGB de píxeles encontrados comparados con los niveles RGB iniciales.

La figura 49 muestra el elemento Iniciar e icono asociado.

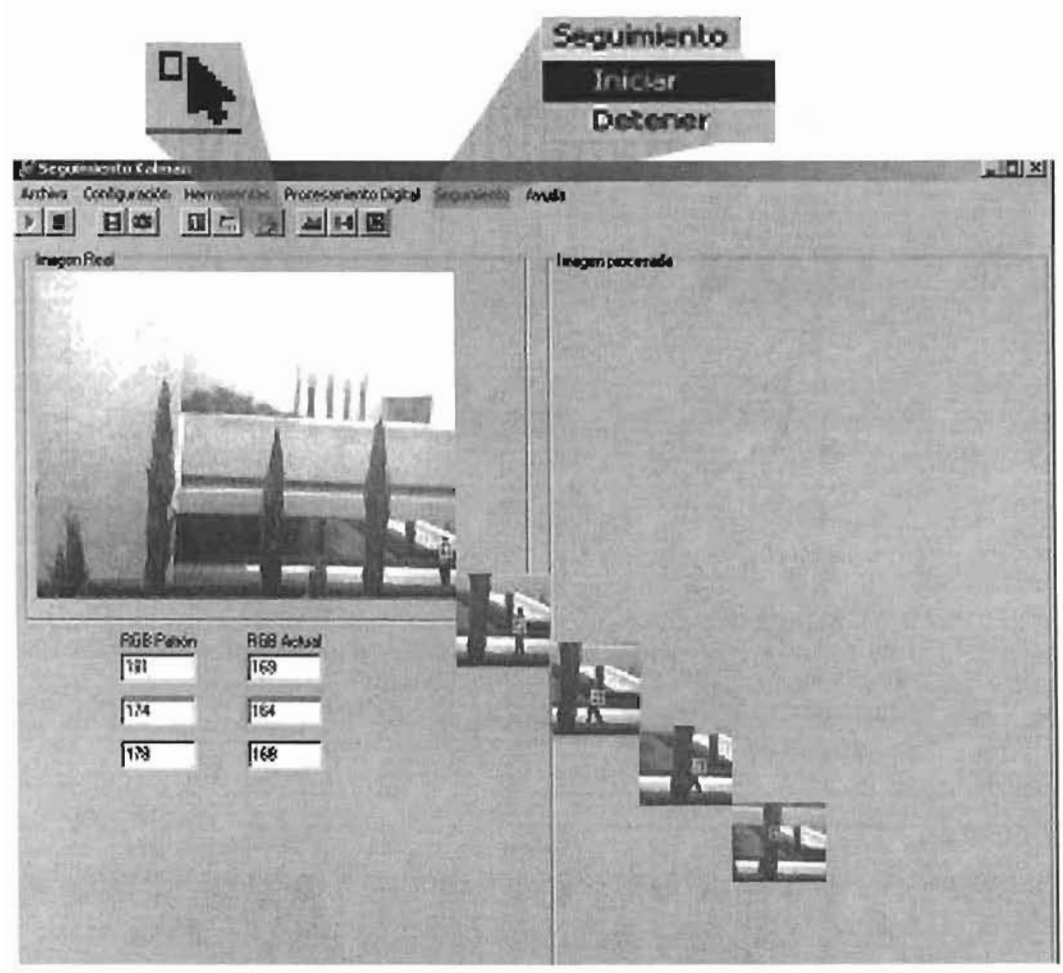


Figura 49 Elemento iniciar e icono asociado

- Despliega la opción Detener. Es habilitada una vez que se ha iniciado el proceso de seguimiento de objeto. Como su nombre lo indica detiene dicho proceso y para la realización de otras tareas. La figura 50 muestra el elemento Seguimiento del menú principal así como los iconos asociados.

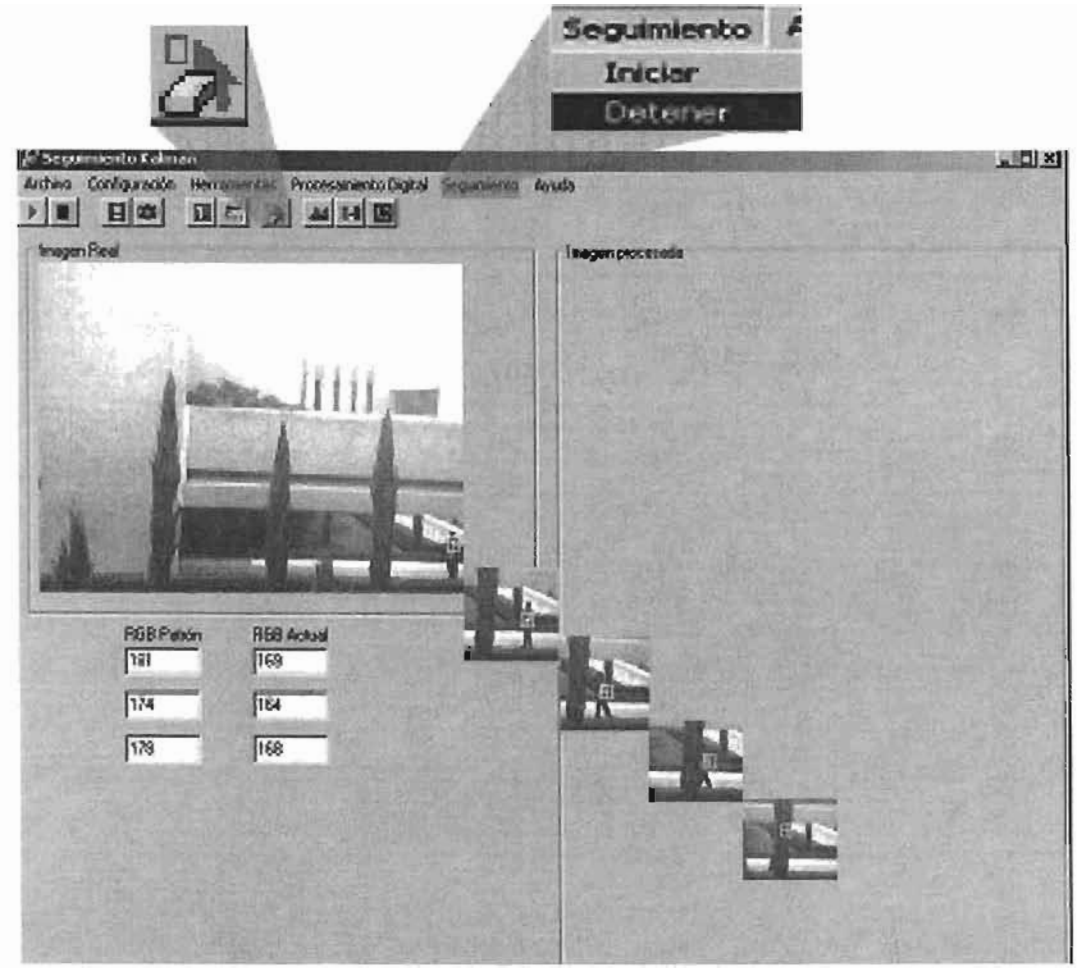


Figura 50 Elemento detener e icono asociado.

- Ayuda
  - Despliega la opción *Ayuda*. Muestra ayuda de forma general acerca del manejo y funcionamiento del sistema así como la versión de éste.

La figura 51 muestra el elemento Ayuda del menú principal

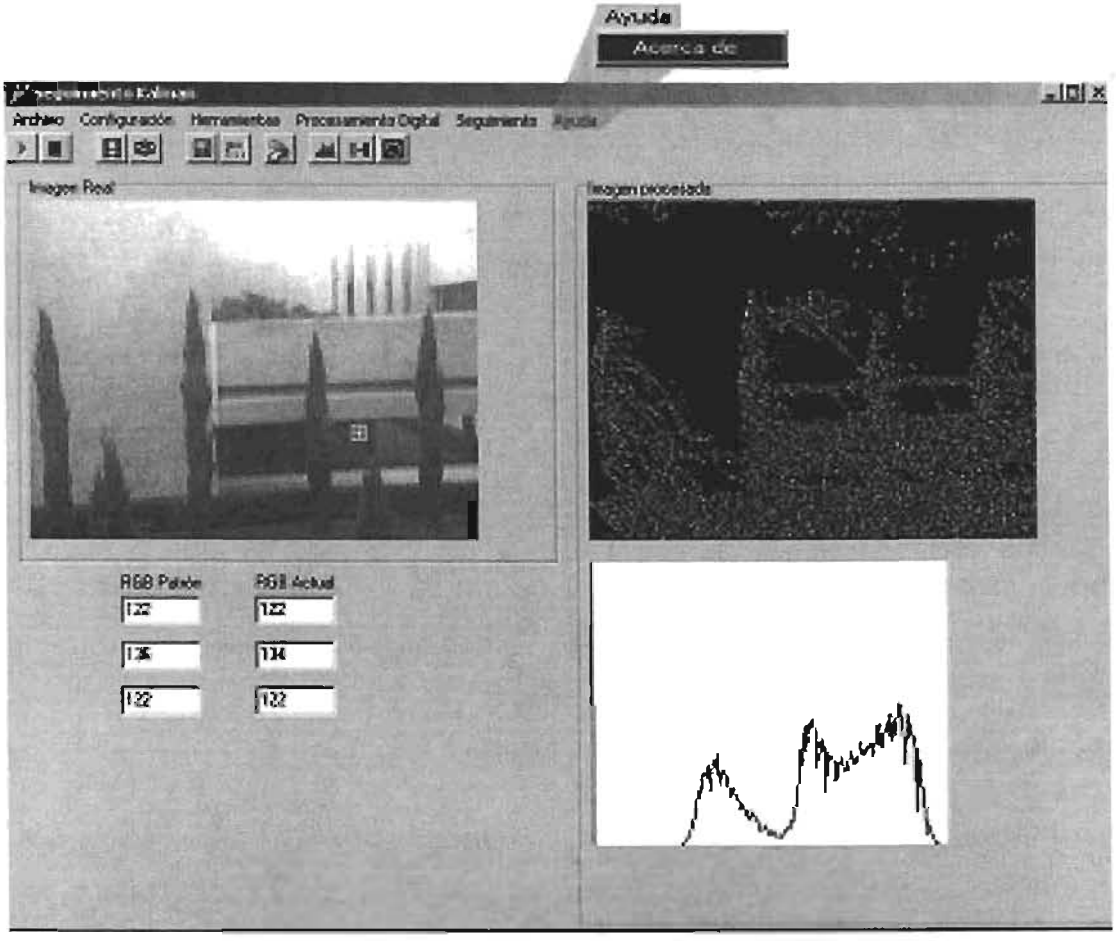


Figura 51 Elemento ayuda.

El desarrollo de la programación fue implementada en el ambiente visual de desarrollo Delphi versión 7.0.

El código fuente se encuentra en el Anexo A.

CONCLUSIONES Y TRABAJO FUTURO

Durante el desarrollo del presente proyecto de tesis se implementaron diferentes módulos con aplicaciones de visión que fueron vitales para la comprensión del funcionamiento del filtro de Kalman utilizado en el seguimiento de objetos.

El desarrollo de este trabajo se centra en dos puntos principales:

- La fiabilidad del filtro de Kalman para la tarea de seguimiento.
  Por las características de funcionamiento e implementación del filtro Kalman se encontró que para la aplicación desarrollada el uso de este filtro fue el adecuado.
- El uso del filtro Kalman para el procesamiento de imagen en tiempo real.
  De forma genérica, el filtro Kalman es utilizado principalmente en aplicaciones en las que necesario corregir mediciones dada la lectura en un elemento de una medición. Utilizar el filtro Kalman de forma recursiva dadas las predicciones de posición y velocidad del mismo, nos mostró que puede ser utilizado como un filtro de tipo predictivo y no sólo correctivo en procesos de tiempo real.

Con respecto a la aplicación de desarrollada se concluye que, por su diseño puede ser utilizada no sólo como sistema de seguimiento sino como un elemento de captura y almacenamiento de imágenes de dispositivos médicos tales como el microscopio para la ayuda de citólogos en el proceso de diagnóstico.

La tarjeta Frame Grabber es de gran utilidad para el proceso de captura de imágenes en tiempo real ya que por su diseño permite un mejor y mayor manejo de los píxeles desplegados, permitiendo desarrollar tareas de procesamiento de imágenes más detalladas; sin embargo, a través del estudio desempeñado se encontró que el utilizar cámaras conectadas por medio del puerto USB dado el tipo de procesamiento establecido, daba el mismo resultado que el utilizar una cámara conectada a la tarjeta Frame Grabber por lo cual se concluye que, el sistema implementado puede ser utilizado en mayor proporción dado el campo tecnológico actual.

Las aplicaciones en el campo de visión en el área de vigilancia son cada día mayormente requeridas, por lo que en el presente trabajo de tesis se brindan las

bases para que trabajos futuros puedan establecerse sobre sistemas de vigilancia basados en conexiones USB logrando con ello una disminución en tiempo de operación y un decremento en costos.

Como línea de investigación futura se propone utilizar el histograma de una imagen y los filtros Sobel para el tratamiento de las imágenes obtenidas en tiempo real , logrando un procesamiento más completo utilizando técnicas basadas en el análisis de textura, detección de bordes y formas, mejorando el proceso de seguimiento así como la implementación de un zoom óptico que proveería a la imagen desplegada una mejor resolución.

Se propone la implementación del algoritmo de búsqueda basado en la c-medias (c-*means*) con lo cual los resultados obtenidos en conjunto por el procesamiento digital de imágenes y el algoritmo del filtro de Kalman daría como resultado la disminución de la probabilidad de perdida del objeto y así como el tiempo de respuesta del sistema.

ANEXO A

CÓDIGO DEL SISTEMA DE SEGUIMIENTO DE OBJETOS MÓVILES UTILIZANDO FILTRO KALMAN

```
//Código Principal

unit USeguimiento_Kalman;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Menus, OleCtrls, ICImagingControl_TLB, Buttons,
  ExtCtrls, ToolWin, ActnMan, ActnCtrls ;

type
  RGB = Array[1..3] of integer;
  RECT=record
    Left : integer;
    Top : integer;
    Right : integer;
    Bottom : integer;
  end;
  TfrmPrincipal = class(TForm)
    MainMenu1: TMainMenu;
    Archivo: TMenuItem;
    IniciaCaptura: TMenuItem;
    PDI: TMenuItem;
    Seguimiento: TMenuItem;
    Histograma: TMenuItem;
    DetieneCaptura: TMenuItem;
    IniciarTrack: TMenuItem;
    DetenerTrack: TMenuItem;
    SBIniciar: TSpeedButton;
    SBParar: TSpeedButton;
```

SBCapturarBMP: TSpeedButton;
SBCapturarAVI: TSpeedButton;
SBConfImagen: TSpeedButton;
SBDispositivo: TSpeedButton;
SD1: TSaveDialog;
SBTracking: TSpeedButton;
SBHist: TSpeedButton;
SBGris: TSpeedButton;
SBSobel: TSpeedButton;
Configuracion: TMenuItem;
ConfiguracionImagen: TMenuItem;
Herramientas: TMenuItem;
CapturaVideo: TMenuItem;
N1: TMenuItem;
Salir: TMenuItem;
ConfiguracionDispositivo: TMenuItem;
CapturaImagen: TMenuItem;
EscalaGrises: TMenuItem;
Sobel: TMenuItem;
Wiener: TMenuItem;
Ayuda: TMenuItem;
AcercaDe: TMenuItem;
Zoom: TMenuItem;
GroupBox1: TGroupBox;
IC1: TICImagingControl;
GroupBox2: TGroupBox;
Imagen1: TImage;
Imagen2: TImage;
Edit1: TEdit;
Edit2: TEdit;
Label1: TLabel;

```
Label2: TLabel;
Edit3: TEdit;
Edit4: TEdit;
Edit5: TEdit;
Edit6: TEdit;
Edit7: TEdit;
Edit8: TEdit;
Edit9: TEdit;
Label3: TLabel;
Label4: TLabel;
Label5: TLabel;
Edit10: TEdit;
Edit11: TEdit;
Edit12: TEdit;
Edit13: TEdit;
Lx_0: TLabel;
Ly_0: TLabel;
Lx_1: TLabel;
Ly_1: TLabel;
Edit14: TEdit;
Edit15: TEdit;
Edit16: TEdit;
Edit17: TEdit;
procedure FormCreate(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure ConfiguracionImagenClick(Sender: TObject);
procedure SBConfImagenClick(Sender: TObject);
procedure SalirClick(Sender: TObject);
procedure ConfiguracionDispositivoClick(Sender: TObject);
procedure SBDispositivoClick(Sender: TObject);
procedure IniciaCapturaClick(Sender: TObject);
```

```
  procedure SBIniciarClick(Sender: TObject);
  procedure DetieneCapturaClick(Sender: TObject);
  procedure SBParar Click(Sender: TObject);
  procedure CapturaVideoClick(Sender: TObject);
  procedure SBCapturarAVIClick(Sender: TObject);
  procedure CapturaImagenClick(Sender: TObject);
  procedure SBCapturarBMPClick(Sender: TObject);
  procedure HistogramaClick(Sender: TObject);
  procedure SBHistClick(Sender: TObject);
  procedure EscalaGrisesClick(Sender: TObject);
  procedure SobelClick(Sender: TObject);
  procedure SBSobelClick(Sender: TObject);
  procedure IniciarTrackClick(Sender: TObject);
  procedure SBTrackingClick(Sender: TObject);
  procedure IC1MouseUp(ASender: TObject; Button, Shift: Integer; XPos,
    YPos: Smallint);
  procedure IC1ImageAvailable(ASender: TObject; BufferIndex: Integer);
private
  { Private declarations }
  //Variables globales
  DisplayBuffer : ImageBuffer;
  UserROI : RECT;
  RegionPatron : RECT;
  RegionActual : RECT;
  UserROICornmited : Boolean;
  IndiceGlobal : integer;
  promw : RGB;
  IBPatron : ImageBuffer;
  ArrPatron : OleVariant;
  IniciaKalman : integer;
  x_0, y_0, x_1, y_1, vx_0, vy_0: integer;
```

```
    //Procedimientos y funciones
    procedure ModoContinuo(BufferIndex : integer; Region : RECT);
    procedure SeguimientoKalman(BufferIndex : integer; Region : RECT);
    procedure DrawRectangleY8(var Arr : OleVariant; Region : RECT;COLOR :
integer);
    procedure HacerConfDisp();
    procedure NormRegBusqueda(Region : RECT; ancho : integer; alto : integer);
    function NormalizaRect(val : RECT):RECT;
    function NivelGrises(var ArregloDisplay : OleVariant; r : RECT): RGB;
  public
    { Public declarations }
    VideoIniciado : Boolean;
    VideoDetenido : Boolean;
  end;
  function encuentra(Prom1:RGB;Prom2:RGB): Boolean;
var
  frmPrincipal: TfrmPrincipal;

implementation

uses Uconfiguracion_imagen, Uconfiguracion, Ucapturar_AVI, UPDI, UKalman;

{$R *.dfm}


{*****************************
* Procedimientos de la forma *
*****************************}

// FormCreate
// Configuración de los valores iniciales
```

```
// cuando se crea por primera vez la forma
procedure TfrmPrincipal.FormCreate(Sender: TObject);
begin
 if IC1.DeviceValid then
 begin
  //Habilita la configuración inicial
  HacerConfDisp();
  //Habilitar botones (Speed Button)
  SBIniciar.Enabled := True;
  SBParar.Enabled := False;
  SBCapturarAVI.Enabled := True;
  SBCapturarBMP.Enabled := True;
  SBDispositivo.Enabled := True;
  SBConfImagen.Enabled := True;
  SBTracking.Enabled := False;
  SBHist.Enabled := True;

  //Deshabilitar Componentes delMenu
  //Archivo
  CapturaImagen.Enabled := True;
  CapturaVideo.Enabled := True;
  Salir.Enabled := True;
  //Vista Previa
  IniciaCaptura.Enabled := True;
  DetieneCaptura.Enabled := False;
  //Configuración
  ConfiguracionDispositivo.Enabled := True;
 end
 else
 begin
  //Deshabilitar botones (Speed Button)
```

```
    SBIniciar.Enabled := False;
    SBParar.Enabled := False;
    SBCapturarAVI.Enabled := False;
    SBCapturarBMP.Enabled := False;
    SBDispositivo.Enabled := True;
    SBConfImagen.Enabled := False;
    SBTracking.Enabled := False;
    SBHist.Enabled := False;

    //Deshabilitar Componentes delMenu
    //Archivo
    CapturaImagen.Enabled := False;
    CapturaVideo.Enabled := False;
    Salir.Enabled := True;
    //Vista Previa
    IniciaCaptura.Enabled := False;
    DetieneCaptura.Enabled := False;
    //Configuración
    ConfiguracionDispositivo.Enabled := True;
  end;
  UserROICommited := False;
end;

procedure TfrmPrincipal.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  if IC1.LiveVideoRunning then
  begin
    if DisplayBuffer.Locked then DisplayBuffer.Unlock;
    frmConfiguracion_imagen.Close;
    IC1.LiveStop
```

```
 end;
end;

{************************************
* Menues y botones de acceso rápido *
************************************}
// ImagenClick
// Abre la forma de configuración de la imagen
// en la cual se puede modificar las características
// de la imagen actual. Dichas características pueden
// ser brillo, contraste, etc.

procedure TfrmPrincipal.ConfiguracionImagenClick(Sender: TObject);
begin
  frmConfiguracion_imagen.IC := IC1;
  frmConfiguracion_imagen.Show;
end;

procedure TfrmPrincipal.SBConfImagenClick(Sender: TObject);
begin
  frmConfiguracion_imagen.IC := IC1;
  frmConfiguracion_imagen.Show;
end;

// SalirClick
// Restauración de valores por defecto
// al cerrar la forma
procedure TfrmPrincipal.SalirClick(Sender: TObject);
begin
  if DisplayBuffer.Locked then DisplayBuffer.Unlock;
  IC1.LiveStop;
```

```
  close;
end;

// DispositivoClick
// Abre la forma de configuración del dispositivo
// en donde se puede realizar la selección del dispositivo
// que se usará para la captura junto con las características
// de canal de entrada, formato de video y norma de video

procedure TfrmPrincipal.ConfiguracionDispositivoClick(Sender: TObject);
begin
  If IC1.LiveVideoRunning then
    IC1.LiveStop;
  frmConfiguracion.IC := IC1;
  frmConfiguracion.ShowModal;

  If IC1.DeviceValid then
  begin
    //Habilita la configuración inicial
    HacerConfDisp();
    //Habilitar botones (Speed Button)
    SBIniciar.Enabled := True;
    SBParar.Enabled := False;
    SBCapturarAVI.Enabled := False;
    SBCapturarBMP.Enabled := False;
    SBDispositivo.Enabled := True;
    SBConfImagen.Enabled := False;
    SBTracking.Enabled := False;

    //Deshabilitar Componentes delMenu
    //Archivo
```

```
    CapturaImagen.Enabled := False;
    CapturaVideo.Enabled := False;
    Salir.Enabled := True;
    //Vista Previa
    IniciaCaptura.Enabled := False;
    DetieneCaptura.Enabled := False;
    //Configuración
    ConfiguracionDispositivo.Enabled := False;
  end;
end;

procedure TfrmPrincipal.SBDispositivoClick(Sender: TObject);
begin
  If IC1.LiveVideoRunning then
    IC1.LiveStop;
  frmConfiguracion.IC := IC1;
  frmConfiguracion.ShowModal;

  If IC1.DeviceValid then
  begin
    //Habilita la configuración inicial
    HacerConfDisp();
    //Habilitar botones (Speed Button)
    SBIniciar.Enabled := True;
    SBParar.Enabled := False;
    SBCapturarAVI.Enabled := False;
    SBCapturarBMP.Enabled := False;
    SBDispositivo.Enabled := True;
    SBConfImagen.Enabled := False;
    SBTracking.Enabled := False;
```

```
    //Deshabilitar Componentes delMenu
    //Archivo
    CapturaImagen.Enabled := False;
    CapturaVideo.Enabled := False;
    Salir.Enabled := True;
    //Vista Previa
    IniciaCaptura.Enabled := False;
    DetieneCaptura.Enabled := False;
    //Configuración
    ConfiguracionDispositivo.Enabled := False;
  end;
end;

// SBIniciarClick
// Al dar click en el botón "Iniciar"
// se comienza la captura de video
procedure TfrmPrincipal.IniciaCapturaClick(Sender: TObject);
begin
  IC1.LiveStart;
  SBTracking.Enabled := True;
  // Almacenar en VideoIniciado que se ha iniciado
  // un video. Si VideoIniciado es verdadero, entonces
  // se asegura que existe una imagen a salvar
  VideoIniciado := TRUE;

    //Habilitar botones (Speed Button)
    SBIniciar.Enabled := False;
    SBParar.Enabled := True;
    SBCapturarAVI.Enabled := True;
    SBCapturarBMP.Enabled := True;
    SBDispositivo.Enabled := True;
```

```
  SBConfImagen.Enabled := True;
  SBTracking.Enabled := True;
  SBHist.Enabled := True;
  SBGris.Enabled := True;
  SBSobel.Enabled := True;

  //Deshabilitar Componentes delMenu
  //Archivo
  CapturaImagen.Enabled := True;
  CapturaVideo.Enabled := True;
  Salir.Enabled := True;
  //Vista Previa
  IniciaCaptura.Enabled := False;
  DetieneCaptura.Enabled := True;
  //Configuración
  ConfiguracionDispositivo.Enabled := True;
end;

procedure TfrmPrincipal.SBIniciarClick(Sender: TObject);
begin
 IC1.LiveStart;
 SBTracking.Enabled := True;
 // Almacenar en VideoIniciado que se ha iniciado
 // un video. Si VideoIniciado es verdadero, entonces
 // se asegura que existe una imagen a salvar
 VideoIniciado := TRUE;

   //Habilitar botones (Speed Button)
   SBIniciar.Enabled := False;
   SBParar.Enabled := True;
   SBCapturarAVI.Enabled := True;
```

```
  SBCapturarBMP.Enabled := True;
  SBDispositivo.Enabled := True;
  SBConfImagen.Enabled := True;
  SBTracking.Enabled := True;
  SBHist.Enabled := True;
  SBGris.Enabled := True;
  SBSobel.Enabled := True;

  //Deshabilitar Componentes delMenu
  //Archivo
  CapturaImagen.Enabled := True;
  CapturaVideo.Enabled := True;
  Salir.Enabled := True;
  //Vista Previa
  IniciaCaptura.Enabled := False;
  DetieneCaptura.Enabled := True;
  //Configuración
  ConfiguracionDispositivo.Enabled := True;
end;

// SBParar Click
// Al dar click en el botón "Parar"
// se detiene la captura de video
procedure TfrmPrincipal.DetieneCapturaClick(Sender: TObject);
begin
 if IC1.LiveVideoRunning then DisplayBuffer.ForceUnlock;
 // Regresar a captura sin el buffer
 IC1.LiveStop;
 if VideoIniciado = true Then
   VideoDetenido := False;
```

```
 //Habilitar botones (Speed Button)
 SBIniciar.Enabled := True;
 SBParar.Enabled := False;
 SBCapturarAVI.Enabled := False;
 SBCapturarBMP.Enabled := False;
 SBDispositivo.Enabled := True;
 SBConfImagen.Enabled := False;
 SBTracking.Enabled := False;
 SBHist.Enabled := False;
 SBGris.Enabled := False;
 SBSobel.Enabled := False;

 //Deshabilitar Componentes delMenu
 //Archivo
 CapturaImagen.Enabled := False;
 CapturaVideo.Enabled := False;
 Salir.Enabled := True;
 //Vista Previa
 IniciaCaptura.Enabled := True;
 DetieneCaptura.Enabled := False;
 //Configuración
 ConfiguracionDispositivo.Enabled := True;
end;

// IniciarClick
// Al acceder a este elemento del menú,
// se da inicio a la captura del video
procedure TfrmPrincipal.HistogramaClick(Sender: TObject);
begin
  IC1.LiveStart;
  SBTracking.Enabled := True;
```

```
// Almacenar en VideoIniciado que se ha iniciado
// un video. Si VideoIniciado es verdadero, entonces
// se asegura que existe una imagen a salvar
VideoIniciado := TRUE;

//Habilitar botones (Speed Button)
  SBIniciar.Enabled := False;
  SBParar.Enabled := True;
  SBCapturarAVI.Enabled := True;
  SBCapturarBMP.Enabled := True;
  SBDispositivo.Enabled := True;
  SBConfImagen.Enabled := True;
  SBTracking.Enabled := True;
  SBHist.Enabled := True;
  SBGris.Enabled := True;
  SBSobel.Enabled := True;

  //Deshabilitar Componentes delMenu
  //Archivo
  CapturaImagen.Enabled := True;
  CapturaVideo.Enabled := True;
  Salir.Enabled := True;
  //Vista Previa
  IniciaCaptura.Enabled := False;
  DetieneCaptura.Enabled := True;
  //Configuración
  ConfiguracionDispositivo.Enabled := True;
end;

procedure TfrmPrincipal.SBPararClick(Sender: TObject);
begin
```

```
if IC1.LiveVideoRunning then DisplayBuffer.ForceUnlock;
// Regresar a captura sin el buffer
IC1.LiveStop;
if VideoIniciado = true Then
  VideoDetenido := False;

//Habilitar botones (Speed Button)
SBIniciar.Enabled := True;
SBParar.Enabled := False;
SBCapturarAVI.Enabled := False;
SBCapturarBMP.Enabled := False;
SBDispositivo.Enabled := True;
SBConfImagen.Enabled := False;
SBTracking.Enabled := False;
SBHist.Enabled := False;
SBGris.Enabled := False;
SBSobel.Enabled := False;

//Deshabilitar Componentes delMenu
//Archivo
CapturaImagen.Enabled := False;
CapturaVideo.Enabled := False;
Salir.Enabled := True;
//Vista Previa
IniciaCaptura.Enabled := True;
DetieneCaptura.Enabled := False;
//Configuración
ConfiguracionDispositivo.Enabled := True;
end;
```

```
// Al dar click a este botón,
// se realiza una llamada a la forma de grabado
// de video tipo AVI
procedure TfrmPrincipal.CapturaVideoClick(Sender: TObject);
begin
  frmEscribir_AVI.IC := IC1;
  frmEscribir_AVI.ShowModal;
end;

procedure TfrmPrincipal.SBCapturarAVIClick(Sender: TObject);
begin
  frmEscribir_AVI.IC := IC1;
  frmEscribir_AVI.ShowModal;
end;

// SBCapturarBMPClick
// Despliega una caja de diálogo y salva la imagen actual
// con el nombre especificado por el usuario
procedure TfrmPrincipal.CapturaImagenClick(Sender: TObject);
begin
  // Revisar si la captura de video ha sido iniciada.
  // Si ya ha sido iniciada existen dos posibles situaciones:
  // 1) La captura de video está en ejecución. En este caso se
  //    captura el frame actual para poder salvarlo
  // 2) La captura de video ha sido detenida. En este caso el
  //    último frame de la captura de video fue obtenido
  //    automáticamente y se puede salvar

  if VideoIniciado = True Then
  begin
    SD1.Filter := 'Bitmap (*.bmp)|*.bmp|';
```

```
    SD1.DefaultExt := '.bmp';
    SD1.Execute;
    IC1.MemorySaveImage(SD1.FileName);
  end;
end;

procedure TfrmPrincipal.SBCapturarBMPClick(Sender: TObject);
begin
// Revisar si la captura de video ha sido iniciada.
  // Si ya ha sido iniciada existen dos posibles situaciones:
  // 1) La captura de video está en ejecución. En este caso se
  //    captura el frame actual para poder salvarlo
  // 2) La captura de video ha sido detenida. En este caso el
  //    último frame de la captura de video fue obtenido
  //    automáticamente y se puede salvar

  if VideoIniciado = True Then
  begin
    SD1.Filter := 'Bitmap (*.bmp)|*.bmp|';
    SD1.DefaultExt := '.bmp';
    SD1.Execute;
    IC1.MemorySaveImage(SD1.FileName);
  end;
end;

procedure TfrmPrincipal.SBHistClick(Sender: TObject);
var
  Bitmap : TBitmap;
  HistCont : array[0..255] of integer;
begin
  Imagen1.Height := IC1.ImageHeight;
```

```
 Imagen1.Width := IC1.Width;
 frmDPI.HImagen1 := Imagen2;
 IC1.MemorySaveImage('C:\WINDOWS\Temp\HistBMP01.bmp');
 Bitmap := TBitmap.create;
 Bitmap.LoadFromFile('C:\WINDOWS\Temp\HistBMP01.bmp');
 Imagen1.Picture.Bitmap := Bitmap;
 frmDPI.Histograma(Bitmap, HistCont);
 frmDPI.Histograma100(HistCont);
 frmDPI.DibujaHist(HistCont);
end;

//Escala de grises
procedure TfrmPrincipal.EscalaGrisesClick(Sender: TObject);
var
  Bitmap : TBitmap;
begin
  if Imagen2.Visible then Imagen2.Visible := False;
  Imagen1.Visible := True;
  Imagen1.Height := IC1.ImageHeight;
  Imagen1.Width := IC1.Width;
  frmDPI.GrisImagen1 := Imagen1;
  IC1.MemorySaveImage('.\GrisBMP01.bmp');
  Bitmap := TBitmap.create;
  Bitmap.LoadFromFile('.\GrisBMP01.bmp');
  Imagen1.Picture.Bitmap := Bitmap;
  frmDPI.RGB_Gris();
  BitMap.SaveToFile('.\Gris01.bmp');
end;

//Sobel
procedure TfrmPrincipal.SobelClick(Sender: TObject);
```

```
var
  Bitmap : TBitmap;
begin
  Imagen1.Height := IC1.ImageHeight;
  Imagen1.Width := IC1.Width;
  frmDPI.GrisImagen1 := Imagen1;
  IC1.MemorySaveImage('C:\SobelBMP01.bmp');
  Bitmap := TBitmap.create;
  Bitmap.LoadFromFile('C:\SobelBMP01.bmp');
  Imagen1.Picture.Bitmap := Bitmap;
  frmDPI.Sobel();
end;

procedure TfrmPrincipal.SBSobelClick(Sender: TObject);
var
  Bitmap : TBitmap;
begin
  Imagen1.Height := IC1.ImageHeight;
  Imagen1.Width := IC1.Width;
  frmDPI.GrisImagen1 := Imagen1;
  IC1.MemorySaveImage('C:\WINDOWS\Temp\SobelBMP01.bmp');
  Bitmap := TBitmap.create;
  Bitmap.LoadFromFile('C:\WINDOWS\Temp\SobelBMP01.bmp');
  Imagen1.Picture.Bitmap := Bitmap;
  frmDPI.Sobel();
end;

//Seguimiento
procedure TfrmPrincipal.IniciarTrackClick(Sender: TObject);
begin
  if not UserROICommited then
```

```
 begin
   UserROICommited := TRUE;
   IniciarTrack.Caption := 'Detener';
 end
 else
 begin
   UserROICommited := FALSE;
   IniciarTrack.Caption := 'Iniciar';
 end;

 // Obtener región patrón
 RegionPatron := NormalizaRect(UserROI);
 IBPatron := IC1.ImageBuffers.Item[IndiceGlobal];
 IBPatron.Lock;
 ArrPatron := IBPatron.GetImageData;
 promw:=NivelGrises(ArrPatron,RegionPatron);
end;

procedure TfrmPrincipal.SBTrackingClick(Sender: TObject);
begin
  if not UserROICommited then
  begin
    UserROICommited := TRUE;
    SBTracking.Glyph.LoadFromFile('.\Bitmaps\ResetROI.bmp');
    SBTracking.Hint := 'Limpiar Region de interés';
    SBTracking.ShowHint := TRUE;
    IniciaKalman := 0;
  end
  else
  begin
    UserROICommited := FALSE;
```

```
  SBTracking.Glyph.LoadFromFile('.\Bitmaps\ROI.bmp');
  SBTracking.Hint := 'Establecer Region de interés';
  SBTracking.ShowHint := TRUE;
  IniciaKalman := 0;
 end;

 RegionPatron := NormalizaRect(UserROI);
 RegionActual := NormalizaRect(UserROI);
 IBPatron := IC1.ImageBuffers.Item[IndiceGlobal];
 IBPatron.Lock;
 ArrPatron := IBPatron.GetImageData;
 promw:=NivelGrises(ArrPatron,RegionPatron);
 Edit1.Text := intToStr(NivelGrises(ArrPatron,RegionPatron)[1]);
 Edit2.Text := intToStr(NivelGrises(ArrPatron,RegionPatron)[2]);
 Edit3.Text := intToStr(NivelGrises(ArrPatron,RegionPatron)[3]);
end;

{*****************************************
* EVENTOS DEL IC CON EL ANILLO DE BUFFERS *
*****************************************}

procedure TfrmPrincipal.IC1MouseUp(ASender: TObject; Button,
  Shift: Integer; XPos, YPos: Smallint);
begin
  if Not (Button = 1) Then
  begin
    UserROI.Left := XPos - 5;
    UserROI.Right := XPos + 5;
    UserROI.Top := IC1.Height - YPos + 5;
    UserROI.Bottom := IC1.ImageHeight - YPos - 5;
```

```
//Inicia seguimiento
if not UserROICommited then
begin
  UserROICommited := TRUE;
  SBTracking.Glyph.LoadFromFile('.\Bitmaps\ResetROI.bmp');
  SBTracking.Hint := 'Limpiar Region de interés';
  SBTracking.ShowHint := TRUE;
  IniciaKalman := 0;
  x_0:=0;
  y_0:=0;
  x_1:=0;
  y_1:=0;
  vx_0:=0;
  vy_0:=0;
 end
else
begin
  UserROICommited := FALSE;
  SBTracking.Glyph.LoadFromFile('.\Bitmaps\ROI.bmp');
  SBTracking.Hint := 'Establecer Region de interés';
  SBTracking.ShowHint := TRUE;
end;

RegionPatron := NormalizaRect(UserROI);
RegionActual := NormalizaRect(UserROI);
IBPatron := IC1.ImageBuffers.Item[IndiceGlobal];
IBPatron.Lock;
ArrPatron := IBPatron.GetImageData;
promw:=NivelGrises(ArrPatron,RegionPatron);
Edit1.Text := intToStr(NivelGrises(ArrPatron,RegionPatron)[1]);
Edit2.Text := intToStr(NivelGrises(ArrPatron,RegionPatron)[2]);
```

```
 Edit3.Text := intToStr(NivelGrises(ArrPatron,RegionPatron)[3]);
 end;
end;

// Image Available
procedure TfrmPrincipal.IC1ImageAvailable(ASender: TObject;
 BufferIndex: Integer);
var
 Region : RECT;
begin
 Region := NormalizaRect(UserROI);
 if not UserROICommited then
   ModoContinuo(BufferIndex,Region)
 else
   SeguimientoKalman(BufferIndex, Region);
end;

//*****************************//
//PROCEDIMIENTOS DE USUARIO.  //
//*****************************//

procedure TfrmPrincipal.ModoContinuo(BufferIndex : integer; Region : RECT);
var
  DisplayArr : OleVariant;
begin
  DisplayBuffer.Unlock;
  DisplayBuffer := IC1.ImageBuffers.Item[IndiceGlobal];
  DisplayBuffer.Lock;
  DisplayArr := DisplayBuffer.GetImageData;
  DrawRectangleY8(DisplayArr,Region,255);
  DisplayBuffer.ReleaseImageData(DisplayArr);
```

```
  IC1.DisplayImageBuffer(DisplayBuffer);
  IndiceGlobal := BufferIndex;
end;

procedure TfrmPrincipal.SeguimientoKalman(BufferIndex : integer; Region :
RECT);
var
  ArrActual : OleVariant;
  ancho, alto : integer;
  encontrado: boolean;
  x_ini, y_ini : integer;
  R2, R3 : RECT;
begin
  //Procesos obligatorios
  DisplayBuffer.Unlock;
  DisplayBuffer := IC1.ImageBuffers.Item[BufferIndex]; //La imagen actual ahora
está en Displaybuffer
  DisplayBuffer.Lock; //Bloqueo de DisplayBuffer para evitar la escritura
  ArrActual := DisplayBuffer.GetImageData; //ArrActual contiene los datos de la
imagen
  //************************************************
  //INICIA BÚSQUEDA
  //BÚSQUEDA LOCAL
  encontrado:=encuentra(NivelGrises(ArrActual,RegionActual), promw);
  x_1:=RegionActual.Left;
  Edit12.Text := IntToStr(x_1);
  y_1:=RegionActual.Top;
  Edit13.Text := IntToStr(y_1);
  vx_0:=x_1-x_0;
  Edit8.Text := IntToStr(vx_0);
  vy_0:=y_1-y_0;
```

```
Edit9.Text := IntToStr(vx_0);
//Muestra RGB
Edit4.Text := intToStr(NivelGrises(ArrActual,RegionActual)[1]);
Edit5.Text := intToStr(NivelGrises(ArrActual,RegionActual)[2]);
Edit6.Text := intToStr(NivelGrises(ArrActual,RegionActual)[3]);
If encontrado then DrawRectangleY8(ArrActual,RegionActual,255)
else
begin
 ancho:= ABS(RegionActual.Right - RegionActual.Left);
 alto := ABS(RegionActual.Top - RegionActual.Bottom);
 //Inicia búsqueda Kalman
 //Aplicación del filtro kalman predictivo
 frmKalman.Kalman(x_0, y_0, vx_0,vy_0,0.1,0.0001,x_1,y_1);
 x_ini := x_1;
 while x_ini < (x_1 + 1) do
 begin
  y_ini := y_1;
  while y_ini < (y_1 + alto) do
  begin
   R3.Left := x_ini;
   R3.Right := x_ini + ancho;
   R3.Top := y_ini;
   R3.Bottom := y_ini + alto;
   encontrado := encuentra(NivelGrises(ArrActual,R3), promw);
   if encontrado then
   begin
    RegionActual := R3;
    DrawRectangleY8(ArrActual,R3,0);
    x_ini := (R2.Right - ancho + 1);
    y_ini := (R2.Bottom);
   end;
```

```
      y_ini := y_ini + alto;
    end;
    x_ini := x_ini + ancho;
  end;
end;

//Si no lo encontró por Kalman
//Inicia búsqueda por procesamiento digital
  R2.Left := RegionActual.Left - ancho;
  R2.Right := RegionActual.Right + ancho;
  R2.Top := RegionActual.Top - alto;
  R2.Bottom := RegionActual.Bottom + alto;
  NormRegBusqueda(R2,ancho,alto);

  x_ini := R2.Left;
  while x_ini < (R2.Right - ancho + 1) do
  begin
    y_ini := R2.Top;
    while y_ini < (R2.Bottom) do
    begin
      R3.Left := x_ini;
      R3.Right := x_ini + ancho;
      R3.Top := y_ini;
      R3.Bottom := y_ini + alto;
      encontrado := encuentra(NivelGrises(ArrActual,R3), promw);
      if encontrado then
      begin
        RegionActual := R3;
        DrawRectangleY8(ArrActual,R3,0);
        x_ini := (R2.Right - ancho + 1);
        y_ini := (R2.Bottom);
```

```
      end;
      y_ini := y_ini + alto;
    end;
    x_ini := x_ini + ancho;
  end;
 end;
 DisplayBuffer.ReleaseImageData(ArrActual);
 IC1.DisplayImageBuffer(DisplayBuffer);
 IndiceGlobal:=BufferIndex;
 IniciaKalman := IniciaKalman + 1;
 Edit7.Text := IntToStr(IniciaKalman);
 x_0 := x_1;
 Edit10.Text := IntToStr(Trunc(x_0));
 y_0 := y_1;
 Edit11.Text := IntToStr(Trunc(y_0));
end;

function encuentra(Prom1:RGB;Prom2:RGB): Boolean;
var
 compara : RGB;
begin
 compara[1] := ABS(prom1[1] - prom2[1]);
 compara[2] := ABS(prom1[2] - prom2[2]);
 compara[3] := ABS(prom1[3] - prom2[3]);
 if (compara[1] > 15) or (compara[2] > 15) or (compara[3] > 15) then
   encuentra := FALSE
 else
   encuentra := TRUE;
end;

function TfrmPrincipal.NormalizaRect(val : RECT):RECT;
```

```
var
  Tmp : integer;
  r : RECT;
begin
  r := val;
  If r.Top > r.Bottom Then
  begin
    Tmp:=r.Top;
    r.Top := r.Bottom;
    r.Bottom := Tmp;
  end;
  If r.Left > r.Right Then
  begin
    Tmp := r.Left;
    r.Left := r.Right;
    r.Right := Tmp;
  end;
  If r.Top < 0 Then
    r.Top := 0;
  If r.Left < 0 Then
    r.Left := 0;
  If r.Bottom >= IC1.ImageHeight Then
    r.bottom := IC1.ImageHeight - 1;
  If r.Right >= IC1.ImageWidth  Then
    r.Right := IC1.ImageWidth  - 1;
  NormalizaRect := r;
end;

procedure TfrmPrincipal.DrawRectangleY8(var Arr : OleVariant; Region :
RECT;COLOR : integer);
var
```

```
  x,y : integer;
  RECT_COLOR : integer;
  xc, yc : integer;
begin
  RECT_COLOR:=COLOR;
  For x:=3*Region.Left To 3*Region.Right do
    Arr[x,Region.Top]:=RECT_COLOR;
  For x:=3*Region.Left To 3*Region.Right do
    Arr[x,Region.Bottom]:=RECT_COLOR;
  For y:=Region.Top To Region.Bottom do
  begin
    Arr[3*Region.Left,y]:=RECT_COLOR;
    Arr[3*Region.Left + 1,y]:=RECT_COLOR;
    Arr[3*Region.Left + 2,y]:=RECT_COLOR;
  end;
  For y:=Region.Top To Region.Bottom do
  begin
    Arr[3*Region.Right,y]:=RECT_COLOR;
    Arr[3*Region.Right + 1,y]:=RECT_COLOR;
    Arr[3*Region.Right + 2,y]:=RECT_COLOR;
  end;
  xc := Region.Left + trunc(ABS((Region.Right - Region.Left))/2);
  yc := Region.Top + trunc(ABS((Region.Top - Region.Bottom))/2);

  if (xc > 3) and (xc < IC1.ImageWidth - 4) then
  for x:= xc-3 to xc+3 do
  begin
    Arr[3*x,yc]:=RECT_COLOR;
    Arr[3*x+1,yc]:=RECT_COLOR;
    Arr[3*x+2,yc]:=RECT_COLOR;
  end;
```

```
 if (yc > 3) and (yc < IC1.Height - 4) then
 begin
   for y:= yc-3 to yc+3 do
   begin
     Arr[3*xc,y]:=RECT_COLOR;
     Arr[3*xc+1,y]:=RECT_COLOR;
     Arr[3*xc+2,y]:=RECT_COLOR;
   end;
 end;
end;

//Procedimientos Auxiliares
{*********************************************************
* HacerConfDisp()
* Configura el ringbuffer, formato de color
* y modo de captura
*********************************************************}
procedure TfrmPrincipal.HacerConfDisp();
begin
  //Tamaño del ringbuffer igual a 5
  IC1.ImageRingBufferSize := 5;

  // Formato de video ICRGB24
  IC1.MemoryCurrentGrabberColorformat := ICRGB24;

  //LiveCaptureContinuous = True hace que cada frame
  //sea copiado al ring buffer
  IC1.LiveCaptureContinuous := TRUE;

  //No guardar la última imagen si es que se llama
```

```
 //a livestop
 IC1.LiveCaptureLastImage := FALSE;

 //Deshabilitar la captura continua. Esto permite desplegar
 //las imágenes desde el ring buffer
 IC1.LiveDisplay := FALSE;

 //Hacer el tamaño del IC del ancho y alto del formato
 //de video actual.
 IC1.Width := IC1.ImageWidth;
 IC1.Height := IC1.ImageHeight;

 DisplayBuffer := IC1.ImageBuffers.Item[1];

 UserROI.Top := 0;
 UserROI.Left := 0;
 UserROI.bottom := 0;
 UserROI.right := 0;
end;

function TfrmPrincipal.NivelGrises(var ArregloDisplay : OleVariant; r : RECT):
RGB;
var
  x,y,NumPixel : integer;
  EscalaRGB : RGB;
begin
  NumPixel := (r.Right - r.Left)*(r.Bottom - r.Top);
  EscalaRGB[1] := 0;
  EscalaRGB[2] := 0;
  EscalaRGB[3] := 0;
  if NumPixel > 0 then
```

```
 begin
   for y:=r.Top to r.Bottom - 2 do
     for x:=r.Left to r.Right - 2 do
     begin
       // Revisar escala de RGB porque se está saliendo
       //*********************************************
       EscalaRGB[1] := EscalaRGB[1] + ArregloDisplay[3*x,y];
       EscalaRGB[2] :=  EscalaRGB[2] + ArregloDisplay[3*x+1,y];
       EscalaRGB[3] := EscalaRGB[3] + ArregloDisplay[3*x + 2,y];
     end;
   NivelGrises[1] := trunc(EscalaRGB[1]/NumPixel);
   NivelGrises[2] := trunc(EscalaRGB[2]/NumPixel);
   NivelGrises[3] := trunc(EscalaRGB[3]/NumPixel);
 end
 else
 begin
   NivelGrises[1] := 0;
   NivelGrises[2] := 0;
   NivelGrises[3] := 0;
 end;
end;

procedure TfrmPrincipal.NormRegBusqueda(Region : RECT; ancho : integer; alto :
integer);
begin
  //ERRORES
    //ERROR LEFT
    if (Region.Left < 0) AND (Region.Right < IC1.ImageWidth - 1)
        AND (Region.Top > 0) AND (Region.Bottom < IC1.ImageHeight - 1) then
    begin
      Region.Left := 0;
```

```
  Region.Right := 3*ancho;
  Region.Top := RegionActual.Top - alto;
  Region.Bottom := RegionActual.Bottom + alto ;
end
//ERROR RIGHT
else if (Region.Left > 0) AND (Region.Right > IC1.ImageWidth - 1)
    AND (Region.Top > 0) AND (Region.Bottom < IC1.ImageHeight - 1) then
begin
  Region.Right := IC1.ImageWidth - 1;
  Region.Left := IC1.ImageWidth - 1 - 3*ancho;
  Region.Top := RegionActual.Top - alto;
  Region.Bottom := RegionActual.Bottom + alto ;
end
//ERROR TOP
else if (Region.Left > 0) AND (Region.Right < IC1.ImageWidth - 1)
    AND (Region.Top < 0) AND (Region.Bottom < IC1.ImageHeight - 1) then
begin
  Region.Top := 0;
  Region.Bottom := 3*alto;
  Region.Left := RegionActual.Left - 1 - ancho;
  Region.Right := RegionActual.Right + ancho;
end
//ERROR BOTTOM
else if (Region.Left > 0) AND (Region.Right < IC1.ImageWidth - 1)
    AND (Region.Top > 0) AND (Region.Bottom > IC1.ImageHeight - 1) then
begin
  Region.Bottom := IC1.ImageHeight - 1;
  Region.Top := IC1.ImageHeight - 1 - 3*alto;
  Region.Left := RegionActual.Left - ancho;
  Region.Right := RegionActual.Right + ancho;
end
```

```
//COMBINADOS
//ERROR LEFT Y BOTTOM
else if (Region.Left < 0) AND (Region.Right < IC1.ImageWidth - 1)
    AND (Region.Top > 0) AND (Region.Bottom > IC1.ImageHeight - 1) then
begin
  Region.Left := 0;
  Region.Bottom := IC1.ImageHeight - 1;
  Region.Top := IC1.ImageHeight - 1 - 3*alto;
  Region.Right := 3*ancho;
end
//ERROR LEFT Y TOP
else if (Region.Left < 0) AND (Region.Right < IC1.ImageWidth - 1)
    AND (Region.Top < 0) AND (Region.Bottom < IC1.ImageHeight - 1) then
begin
  Region.Left := 0;
  Region.Top := 0;
  Region.Bottom := 3*alto;
  Region.Right := 3*ancho;
end
//ERROR RIGHT Y BOTTOM
else if (Region.Left > 0) AND (Region.Right > IC1.ImageWidth - 1)
    AND (Region.Top > 0) AND (Region.Bottom > IC1.ImageHeight - 1) then
begin
  Region.Bottom := IC1.ImageHeight - 1;
  Region.Right := IC1.ImageWidth - 1;
  Region.Left := IC1.ImageWidth - 1 - 3*ancho;
  Region.Top := IC1.ImageHeight - 1 - 3*alto;
end
//ERROR RIGHT Y TOP
else if (Region.Left > 0) AND (Region.Right > IC1.ImageWidth - 1)
    AND (Region.Top < 0) AND (Region.Bottom < IC1.ImageHeight - 1) then
```

```
   begin
     Region.Right := IC1.ImageWidth - 1;
     Region.Top := 0;
     Region.Left := IC1.ImageWidth - 1 - 3*ancho;
     Region.Bottom := 3*alto;
   end;
end;
end.

//Unidad de Seguimiento Kalman

program PSeguimiento_Kalman;

uses
  Forms,
  USeguimiento_Kalman in 'USeguimiento_Kalman.pas' {frmPrincipal},
  UPDI in '..\Seguimiento Nivel RGB Original P1\Procesamiento digital de
imágenes\UPDI.pas' {frmDPI},
  Uconfiguracion_imagen in '..\Seguimiento Nivel RGB Original
P1\Configurar_imagen\Uconfiguracion_imagen.pas' {frmConfiguracion_imagen},
  Uconfiguracion in '..\Seguimiento Nivel RGB Original
P1\Configuracion_dispositivo\Uconfiguracion.pas' {frmConfiguracion},
  Ucapturar_AVI in '..\Seguimiento Nivel RGB Original
P1\Capturar_AVI\Ucapturar_AVI.pas' {frmEscribir_AVI},
  UKalman in '..\Seguimiento Nivel RGB Original P2\Kalman\UKalman.pas'
{frmKalman};

{$R *.res}

 begin
   Application.Initialize;
```

```
  Application.CreateForm(TfrmPrincipal, frmPrincipal);
  Application.CreateForm(TfrmDPI, frmDPI);
  Application.CreateForm(TfrmConfiguracion_imagen, frmConfiguracion_imagen);
  Application.CreateForm(TfrmConfiguracion, frmConfiguracion);
  Application.CreateForm(TfrmEscribir_AVI, frmEscribir_AVI);
  Application.CreateForm(TfrmKalman, frmKalman);
  Application.Run;
end.


//Unidad Kalman

unit UKalman;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs,MtxVec;

type
  TfrmKalman = class(TForm)
  private
    { Private declarations }
  public
    { Public declarations }
    //XPosF, YPosF : integer;
    procedure Kalman(x_0 : integer; y_0 : integer; vx_0 : integer; vy_0 : integer;
 varR : double; varQ : double; var x_1:integer; var y_1:integer);
  end;
```

```
var
 frmKalman: TfrmKalman;

implementation

{$R *.dfm}

procedure ZerosMat(a : TMtx; Rows : integer; Cols : integer);overload;
var
 i,j : integer;
begin
 for i:=0 to a.Rows - 1 do
  for j:=0 to a.Cols - 1 do
   a.Values[i,j] := 0;
end;

procedure ZerosMat(a : TMtx);overload;
var
 i,j : integer;
begin
 for i:=0 to a.Rows - 1 do
  for j:=0 to a.Cols - 1 do
   a.Values[i,j] := 0;
end;

procedure TfrmKalman.Kalman(x_0 : integer; y_0 : integer; vx_0 : integer; vy_0 :
integer; varR : double; varQ : double; var x_1:integer; var y_1:integer);
var
 H_k, H_kT, A_k1, I, R_k, Q_k1, S_pk, P_k, S_pk1, P_k1,P_pk, K_k, Z_k, K_kt:
TMtx;
 a,b,c,d,e,f,g,h : TMtx;
```

```
  vx_k, vy_k, x_k, y_k, zx_k, zy_k : integer;
  x_fin, y_fin : integer;
begin
  CreateIt(H_k,A_k1,R_k,Q_k1);
  CreateIt(S_pk, P_k, P_pk, S_pk1);
  CreateIt(P_k1, P_pk, K_k, H_kT);
  CreateIt(Z_k,I,K_kt);
  CreateIt(a,b,c,d);
  CreateIt(e,f);

  //Tamaño de las matrices
  H_k.Size(2,4,false);
  H_kT.Size(4,2,false);
  A_k1.Size(4,4,false);
  I.Size(4,4,false);
  R_k.Size(2,2,false);
  Q_k1.Size(4,4,false);
  S_pk.Size(4,1,false);
  P_k.Size(4,4,false);
  P_k1.Size(4,4,false);
  S_pk1.Size(4,1,false);
  P_pk.Size(4,4,false);
  K_k.Size(4,2,false);
  Z_k.Size(2,1,false);

   //Matriz de mediciones
  ZerosMat(H_k,2,4);
  H_k.Values[0,0]:=1;
  H_k.Values[1,1]:=1;

  ZerosMat(H_kT,4,2);
```

```
H_kT.Values[0,0]:=1;
H_kT.Values[1,1]:=1;

//Matriz de transición de estados
ZerosMat(A_k1,4,4);
A_k1.Values[0,0]:=1;
A_k1.Values[0,2]:=1;
A_k1.Values[1,1]:=1;
A_k1.Values[1,3]:=1;
A_k1.Values[2,2]:=1;
A_k1.Values[3,3]:=1;

//Matriz identidad
ZerosMat(I,4,4);
I.Values[0,0]:=1;
I.Values[1,1]:=1;

//Pseudovariables (R de k-1 // Q de k-1)
ZerosMat(R_k,2,2);
R_k.Values[0,0] := varR;
R_k.Values[1,1] := varR;

ZerosMat(Q_k1,4,4);
Q_k1.Values[0,0] := 1;
Q_k1.Values[1,1] := 1;
Q_k1.Values[2,2] := 1;
Q_k1.Values[3,3] := 1;

//Valores iniciales (P_0 // S_0)
vx_k := vx_0;
vy_k := vy_0;
```

```
x_k := x_0 + vx_k;
zx_k := x_0 + vx_k;
y_k := y_0 + vy_k;
zy_k := y_0 + vy_k;

S_pk.Values[0,0]:=x_k;
S_pk.Values[1,0]:=y_k;
S_pk.Values[2,0]:=vx_k;
S_pk.Values[3,0]:=vy_k;

ZerosMat(P_k,4,4);
P_k.Values[0,0]:=1;
P_k.Values[1,1]:=1;
P_k.Values[2,2]:=1;
P_k.Values[3,3]:=1;

//Matrices auxiliares
CreateIt(a,b,c,d);
CreateIt(e,f,g,h);

//Iteraciones de Kalman
try
  //Actualización estados k con estado k-1
  S_pk1 := S_pk;
  P_k1 := P_k;
  //Matriz de mediciones
  Z_k.Values[0,0]:=zx_k;
  Z_k.Values[1,0]:=zy_k;

  //PRIMERA ECUACIÓN KALMAN
  P_pk.Add(a.Mul(A_k1, b.Mul(P_k1,A_k1.Transp)),Q_k1);
```

```
//SEGUNDA ECUACIÓN KALMAN
ZerosMat(a);
ZerosMat(b);
ZerosMat(c);
ZerosMat(d);
a.Mul(P_pk, H_kT);
b.Add(c.Mul(d.Mul(H_k,P_pk),H_kT),R_k);
K_k.Mul(a,b.Inv);

//TERCERA ECUACIÓN KALMAN
ZerosMat(a);
ZerosMat(b);
ZerosMat(c);
ZerosMat(d);

S_pk.Add(a.Mul(A_k1,S_pk1),b.Mul(K_k,c.Sub(d.Mul(H_k,e.Mul(A_k1,S_pk1)),Z_k
)));

//CUARTA ECUACIÓN KALMAN
ZerosMat(a);
ZerosMat(b);
ZerosMat(c);
ZerosMat(d);
ZerosMat(e);
ZerosMat(f);
k_kt.Transp(K_k);
a.Sub(f.Mul(K_k,H_k),I);
b.Transp(a);
P_k.Sub(e.Mul(K_k,f.Mul(R_k,K_kt)),c.Mul(a,d.Mul(P_pk,b)));
```

```
    //Limpiar matrices auxiliares
    ZerosMat(a);
    ZerosMat(b);
    ZerosMat(c);
    ZerosMat(d);
    ZerosMat(e);

    x_fin:= round(S_pk[0,0]);
    y_fin:=round(S_pk[1,0]);
  finally
    FreeIt(H_k,A_k1,R_k,Q_k1);
    FreeIt(S_pk, P_k, P_pk, S_pk1);
    FreeIt(P_k1, P_pk, K_k, H_kT);
    FreeIt(Z_k,I,K_kt);
    FreeIt(a,b,c,d);
    FreeIt(e,f);
  end;
end;
end.

//Unidad Configuración de dispositivo

unit Uconfiguracion;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ICImagingControl_TLB, StdCtrls;

type
```

```
TfrmConfiguracion = class(TForm)
  lblFuente: TLabel;
  lblNorma: TLabel;
  lblFormato: TLabel;
  lblCanal: TLabel;
  btnAceptar: TButton;
  btnCancelar: TButton;
  cboFuente_Video: TComboBox;
  cboNorma_Video: TComboBox;
  cboFormato_Video: TComboBox;
  cboCanal_Entrada: TComboBox;
  procedure btnCancelarClick(Sender: TObject);
  procedure btnAceptarClick(Sender: TObject);
  procedure cboFuente_VideoClick(Sender: TObject);
  procedure cboNorma_VideoClick(Sender: TObject);
  procedure cboFormato_VideoClick(Sender: TObject);
  procedure cboCanal_EntradaClick(Sender: TObject);
  procedure FormShow(Sender: TObject);
 private
  { Private declarations }
  procedure ActualizaDispositivos();
  procedure ActualizaNormasVideo();
  procedure ActualizaFormatosVideo();
  procedure ActualizaCanalesEntrada();
 public
  { Public declarations }
  IC: TICImagingControl;
  Cancelado : boolean;
 end;

var
```

```
  frmConfiguracion: TfrmConfiguracion;

implementation

{$R *.dfm}
{*******************************************************
* EVENTOS DE LA FORMA
*******************************************************}

{*******************************************************
* FormShow
* Llena el combo de fuentes de video con todos los dispositivos
* de captura de video disponibles y elegí el primero de la lista.
* Esto disparará un evento de click en el combo de fuentes de video
* y abrirá el dispositivo
*******************************************************}
procedure TfrmConfiguracion.FormShow(Sender: TObject);
begin
  cancelado := FALSE;
  ActualizaDispositivos();
  cboFuente_VideoClick(Sender);
end;
{*******************************************************
*  BOTONES
*******************************************************}

{*******************************************************
* btnCancelarClick
* Cierra la forma y actualiza el valor de cancelado a TRUE
*******************************************************}
```

```
procedure TfrmConfiguracion.btnCancelarClick(Sender: TObject);
begin
  cancelado := TRUE;
  close;
end;

{*******************************************************
* btnAceptarClick
* Cierra la forma.
*******************************************************}
procedure TfrmConfiguracion.btnAceptarClick(Sender: TObject);
begin
  close;
end;

{*******************************************************
*  COMBOS
*******************************************************}

{*******************************************************
* cboFuenteChange
* Obtiene las entradas y formados de video del dispositivo
* seleccionado y meterá la información en los combos respectivos
*******************************************************}
procedure TfrmConfiguracion.cboFuente_VideoClick(Sender: TObject);
begin
  try
    if IC.Devices.Count > 0 Then
    begin
      //Abrir el dispositivo
      IC.Device := cboFuente_Video.Text
```

```
  end;

  //Obtener formas de video, formatos y entradas
  ActualizaNormasVideo();
  ActualizaCanalesEntrada();
  ActualizaFormatosVideo();
 except
  IC.Device := '';
  cboNorma_Video.Enabled := FALSE;
  cboNorma_Video.Clear;
  cboNorma_Video.Text := 'n/a';

  cboCanal_Entrada.Enabled := FALSE;
  cboCanal_Entrada.Clear;
  cboCanal_Entrada.Text := 'n/a';

  cboFormato_Video.Enabled := FALSE;
  cboFormato_Video.Clear;
  cboFormato_Video.Text := 'n/a';

  messagedlg('El dispositivo seleccionado no es válido',mtError,[mbOk],0);
 end;
end;

{*****************************************************
* cboNorma_VideoClick
* Selecciona una norma de video
*****************************************************}
procedure TfrmConfiguracion.cboNorma_VideoClick(Sender: TObject);
begin
 if cboNorma_Video.Enabled then
```

```
   IC.VideoNorm := cboNorma_Video.Text;
  ActualizaFormatosVideo();
end;

{*****************************************************
* cboCanal_EntradaClick
* Selecciona un canal de entrada
*****************************************************}
procedure TfrmConfiguracion.cboCanal_EntradaClick(Sender: TObject);
begin
if cboCanal_Entrada.Enabled then
   IC.InputChannel := cboCanal_Entrada.Text;
end;

{*****************************************************
* cboFormato_VideoClick
* Selecciona un formato de video
*****************************************************}
procedure TfrmConfiguracion.cboFormato_VideoClick(Sender: TObject);
begin
  if IC.DeviceValid then
    IC.VideoFormat := cboFormato_Video.Text;
end;

{*****************************************************
* PROCEDIMIENTOS AUXILIARES
*****************************************************}

{*****************************************************
*  ActualizaDispositivos()
* Intenta encontrar todos los dispositivos de captura
```

```
* disponibles y los despliega en un combo llamado
* cboFuente_Video
*******************************************************}
procedure TfrmConfiguracion.ActualizaDispositivos();
var
  nombre : WideString;
  idx,i : integer;
  ColeccionDispositivos : Devices;
begin
  idx := -1;
  ColeccionDispositivos := IC.Devices;
  //Si se ha seleccionado con anterioridad, obtener su nombre
  //para seleccionarlo después en el combo
  if IC.DeviceValid = TRUE then
  begin
    nombre := IC.Device;

    //Obtener el índice del dispositivo en la colección
    idx := ColeccionDispositivos.FindIndex(nombre);
  end;
  cboFuente_Video.Clear;
  if ColeccionDispositivos.Count > 0 Then
  begin
    //Si encontraron dispositivos añadirlos al combo
    for i:=0 to IC.Devices.Count - 1 do
      cboFuente_Video.Items.Add(ColeccionDispositivos.Item[i].Name);
    cboFuente_Video.Enabled := TRUE;
    if idx <> -1 Then
      cboFuente_Video.ItemIndex := idx - 1
    else
      cboFuente_Video.ItemIndex := 0;
```

```
 end
 else
 begin
   //No se han hallado dispositivos
   cboFuente_Video.Items.Add('No se hallaron dispositivos');
   cboFuente_Video.Enabled := FALSE;
   cboFuente_Video.ItemIndex := 0;
 end;
end;

{*****************************************************
*  ActualizaNormasVideo()
* Intenta encontrar todos las normas de video del
* dispositivo seleccionado y las despliega en el combo
* llamado cboNormas_Video
*****************************************************}
procedure TfrmConfiguracion.ActualizaNormasVideo();
var
  nombre : WideString;
  VidNrmCol : VideoNorms;
  idx, i : integer;
begin
  idx := -1;
  VidNrmCol := IC.VideoNorms;

  cboNorma_Video.Clear;
  if VidNrmCol.Count > 0 Then
  begin
    //Si se han encontrado normas de video
    //añadirlas al combo
    for i:=0 to VidNrmCol.Count - 1 do
```

```
    cboNorma_Video.Items.Add(VidNrmCol.Item[i].Name);

   //Obtiene la norma de video previamente seleccionada
   cboNorma_Video.Enabled := TRUE;
   nombre := IC.VideoNorm;

   idx := VidNrmCol.FindIndex(nombre);
   if idx <> -1 Then
     cboNorma_Video.ItemIndex := idx - 1
   else
     cboNorma_Video.ItemIndex := 0;
 end
 else
 begin
   //No se hallaron normas de video
   cboNorma_Video.Items.Add('n/a');
   cboNorma_Video.Enabled := FALSE;
   cboNorma_Video.ItemIndex := 0;
 end;
end;

{*******************************************************
*  ActualizaFormatosVideo()
* Intenta encontrar todos los formatos de video del
* dispositivo seleccionado y las despliega en el combo
* llamado cboFormatos_Video
*******************************************************}
procedure TfrmConfiguracion.ActualizaFormatosVideo();
var
  nombre : WideString;
  VidFmtCol : VideoFormats;
```

```
  idx, i : integer;
begin
  idx := 0;
  VidFmtCol := IC.VideoFormats;

  if VidFmtCol.Count > 0 Then
  begin
    //Si se han encontrado formatos de video
    //añadirlos al combo
    cboFormato_Video.Clear;
    for i:=1 to VidFmtCol.Count do
      cboFormato_Video.Items.Add(VidFmtCol.Item[i].Name);

    //Obtiene el formato de video previamente seleccionado
    cboFormato_Video.Enabled := TRUE;
    nombre := IC.VideoFormat;

    idx := VidFmtCol.FindIndex(nombre);
    if idx <> -1 Then
      cboFormato_Video.ItemIndex := idx - 1
    else
      cboFormato_Video.ItemIndex := 0;
  end
  else
  begin
    //No se hallaron formatos de video
    cboFormato_Video.Items.Add('n/a');
    cboFormato_Video.Enabled := FALSE;
    cboFormato_Video.ItemIndex := 0;
  end;
end;
```

```
{*******************************************************
*  ActualizaCanalesEntrada()
* Intenta encontrar todos los canales de entrada del
* dispositivo seleccionado y las despliega en el combo
* llamado cboCanales_Entrada
*******************************************************}
procedure TfrmConfiguracion.ActualizaCanalesEntrada();
var
  nombre : WideString;
  ColeccionCanales : InputChannels;
  idx, i : integer;
begin
  idx := -1;
  ColeccionCanales := IC.InputChannels;

  cboCanal_Entrada.Clear;
  if ColeccionCanales.Count > 0 Then
  begin
    //Si se hallaron canales de entrada agregarlos al combo
    for i:=0 to ColeccionCanales.Count - 1 do
      cboCanal_Entrada.Items.Add(ColeccionCanales.Item[i].Name);
    nombre := IC.InputChannel;
    idx := ColeccionCanales.FindIndex(nombre);
    if idx <> -1 Then
      cboCanal_Entrada.ItemIndex := idx - 1
    else
      cboCanal_Entrada.ItemIndex := 0;
    cboCanal_Entrada.Enabled := TRUE;
  end
  else
```

```
  begin
    //No se han hallado canales de entrada
    cboCanal_Entrada.Items.Add('n/a');
    cboCanal_Entrada.Enabled := FALSE;
    cboCanal_Entrada.ItemIndex := 0;
  end;
end;
end.

//Unidad de Captura de Video

unit Uescribir_AVI;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, OleCtrls, ICImagingControl_TLB;

type
  TfrmCapturar_AVI = class(TForm)
    IC1: TICImagingControl;
    btnIniciar: TButton;
    btnDetener: TButton;
    btnCapturar_AVI: TButton;
    procedure btnCapturar_AVIClick(Sender: TObject);
    procedure btnIniciarClick(Sender: TObject);
    procedure btnDetenerClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
```

```
  public
    { Public declarations }
  end;

var
  frmCapturar_AVI: TfrmCapturar_AVI;

implementation

uses Ucapturar_AVI;

{$R *.dfm}

procedure TfrmCapturar_AVI.btnCapturar_AVIClick(Sender: TObject);
begin
  frmEscribir_AVI.IC := IC1;
  frmEscribir_AVI.ShowModal;
end;



procedure TfrmCapturar_AVI.btnIniciarClick(Sender: TObject);
begin
  IC1.LiveStart;
end;



procedure TfrmCapturar_AVI.btnDetenerClick(Sender: TObject);
begin
  IC1.LiveStop;
```

```
end;



procedure TfrmCapturar_AVI.FormCreate(Sender: TObject);
begin
 if IC1.DeviceValid then
 begin
   btnIniciar.Enabled := TRUE;
   btnDetener.Enabled := TRUE;
   btnCapturar_AVI.Enabled := TRUE;
   //Deshabilita el tamaño por defecto
   IC1.LiveDisplayDefault := FALSE;
   IC1.LiveDisplayHeight := IC1.Height;
   IC1.LiveDisplayWidth := IC1.Width;
 end
 else
 begin
   btnIniciar.Enabled := FALSE;
   btnDetener.Enabled := FALSE;
   btnCapturar_AVI.Enabled := FALSE;
   messagedlg('No se ha seleccionado el dispositivo',mtError,[mbOk],0);
 end;
end;

end.


//Unidad de Configuración de Imagen

unit Uconfiguracion_imagen;
```

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ComCtrls, StdCtrls, ICImagingControl_TLB;

type
  TfrmConfiguracion_imagen = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    Label15: TLabel;
    Label16: TLabel;
    Label17: TLabel;
    lblTasas_Frame: TLabel;
    TrackBar2: TTrackBar;
    TrackBar7: TTrackBar;
    TrackBar8: TTrackBar;
    TrackBar9: TTrackBar;
    TrackBar10: TTrackBar;
```

TrackBar11: TTrackBar;
TrackBar12: TTrackBar;
TrackBar13: TTrackBar;
TrackBar14: TTrackBar;
TrackBar15: TTrackBar;
TrackBar16: TTrackBar;
TrackBar17: TTrackBar;
TrackBar1: TTrackBar;
TrackBar3: TTrackBar;
TrackBar4: TTrackBar;
TrackBar5: TTrackBar;
TrackBar6: TTrackBar;
CheckBox16: TCheckBox;
btnCerrar: TButton;
cboTasas_Frame: TComboBox;
CheckBox1: TCheckBox;
CheckBox2: TCheckBox;
CheckBox3: TCheckBox;
CheckBox4: TCheckBox;
CheckBox5: TCheckBox;
CheckBox6: TCheckBox;
CheckBox7: TCheckBox;
CheckBox8: TCheckBox;
CheckBox9: TCheckBox;
CheckBox10: TCheckBox;
CheckBox11: TCheckBox;
CheckBox12: TCheckBox;
CheckBox13: TCheckBox;
CheckBox14: TCheckBox;
CheckBox15: TCheckBox;
CheckBox17: TCheckBox;

```
  procedure btnCerrarClick(Sender: TObject);
  procedure FormShow(Sender: TObject);
  procedure TrackBar1Change(Sender: TObject);
  procedure CheckBox1Click(Sender: TObject);
  procedure cboTasas_FrameClick(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
  IC: TICImagingControl;
  procedure SetSlidersAndCheckboxes();
  procedure InitSliderAndCheckbox(IndicePropiedad : integer;
                                  ConfiguracionDisponible : boolean;
                                  RangoSlider : OleVariant;
                                  ValorSlider : integer;
                                  AutoDisponible : Boolean;
                                  AutoValor : boolean);
  procedure SetSlider(IndicePropiedad : integer;
                      ConfiguracionDisponible : boolean;
                      RangoSlider : OleVariant;
                      ValorSlider : integer);
  procedure SetCheckbox(IndicePropiedad : integer;
                        AutoDisponible : boolean;
                        AutoValor : boolean);
  procedure ObtenerTasasFrameDisponibles();
  procedure AlActualizarStartStop();
  procedure HabilitaControlesTasasFrame(Disponible : boolean);
 end;
var
 frmConfiguracion_imagen: TfrmConfiguracion_imagen;
```

```
implementation

{$R *.dfm}

{*********************************************************
* FormShow
* Inicializa las barras deslizadoras y los botones de
* chequeo de acuerdo con las propiedades disponibles
*********************************************************}
procedure TfrmConfiguracion_imagen.FormShow(Sender: TObject);
begin
  try
    SetSlidersAndCheckboxes();
  except
    Messagedlg('Error',mtError,[mbOk],0);
  end;
end;

{*********************************************************
* btnCerrarClick
* Cierra la ventana
*********************************************************}
procedure TfrmConfiguracion_imagen.btnCerrarClick(Sender: TObject);
begin
  close;
end;

{*********************************************************
* SetSlidersAndCheckboxes
* Procedimiento que Habilita/Deshabilita barras deslizables
* y botones de chequeo de acuerdo a las propiedades del
```

```
* dispositivo seleccionadas.
*******************************************************}
procedure TfrmConfiguracion_imagen.SetSlidersAndCheckboxes();
begin
  if IC.DeviceValid then
  begin
    //Brillo
    if IC.BrightnessAvailable then
    begin
      if IC.BrightnessAutoAvailable then
      begin

InitSliderAndCheckbox(1,TRUE,IC.BrightnessRange,IC.Brightness,TRUE,IC.Bright
nessAuto);
      end
      else
      begin

InitSliderAndCheckbox(1,IC.BrightnessAvailable,IC.BrightnessRange,IC.Brightnes
s,False, False);
      end;
    end
    else
    begin
      InitSliderAndCheckbox(1,False,0,0,False,False);
    end;

    //Contraste
    if IC.ContrastAvailable then
    begin
      if IC.ContrastAutoAvailable then
```

```
      begin
        InitSliderAndCheckbox(2,True, IC.ContrastRange,
IC.Contrast,TRUE,IC.ContrastAuto);
      end
      else
      begin

InitSliderAndCheckbox(2,IC.ContrastAvailable,IC.ContrastRange,IC.Contrast,False
, False);
      end;
    end
    else
    begin
      InitSliderAndCheckbox(2,False,0,0,False,False);
    end;

    //Matiz
    if IC.HueAvailable then
    begin
      if IC.HueAutoAvailable then
      begin
        InitSliderAndCheckbox(3,True,IC.HueRange,IC.Hue,True,IC.HueAuto);
      end
      else
      begin
        InitSliderAndCheckbox(3,IC.HueAvailable,IC.HueRange,IC.Hue,False,False);
      end;
    end
    else
    begin
      InitSliderAndCheckbox(3, False, 0, 0, False, False);
```

```
    end;

    //Saturación
    if IC.SaturationAvailable then
    begin
      if IC.SaturationAutoAvailable then
      begin

InitSliderAndCheckbox(4,True,IC.SaturationRange,IC.Saturation,True,IC.Saturatio
nAuto);
      end
      else
      begin

InitSliderAndCheckbox(4,IC.SaturationAvailable,IC.SaturationRange,IC.Saturation,
False, False);
      end;
    end
    else
    begin
      InitSliderAndCheckbox(4, False, 0, 0, False, False);
    end;

    //Forma
    if IC.SharpnessAvailable then
    begin
      if IC.SharpnessAutoAvailable then
      begin

InitSliderAndCheckbox(5,True,IC.SharpnessRange,IC.Sharpness,True,IC.Sharpne
ssAuto);
```

```
    end
    else
    begin

InitSliderAndCheckbox(5,IC.SharpnessAvailable,IC.SharpnessRange,IC.Sharpnes
s,False,False);
    end;
  end
  else
  begin
    InitSliderAndCheckbox(5, False, 0, 0, False, False);
  end;

  //Gama
  if IC.GammaAvailable then
  begin
    if IC.GammaAutoAvailable Then
    begin

InitSliderAndCheckbox(6,True,IC.GammaRange,IC.Gamma,True,IC.GammaAuto);
    end
    else
    begin

InitSliderAndCheckbox(6,IC.GammaAvailable,IC.GammaRange,IC.Gamma,False,
False);
    end;
  end
  else
  begin
    InitSliderAndCheckbox(6, False, 0, 0, False, False);
```

```
    end;

    //Balance blanco
    if IC.WhiteBalanceAvailable then
    begin
      if IC.WhiteBalanceAutoAvailable Then
      begin

InitSliderAndCheckbox(7,True,IC.WhiteBalanceRange,IC.WhiteBalance,True,IC.W
hiteBalanceAuto);
      end
      else
      begin

InitSliderAndCheckbox(7,IC.WhiteBalanceAvailable,IC.WhiteBalanceRange,IC.Whi
teBalance,False, False);
      end;
    end
    else
    begin
      InitSliderAndCheckbox(7, False, 0, 0, False, False);
    end;

    //Compensación de iluminación posterior
    if IC.BacklightCompensationAvailable then
    begin
      if IC.BacklightCompensationAutoAvailable Then
      begin
        InitSliderAndCheckbox(8,
True,IC.BacklightCompensationRange,IC.BacklightCompensation,True,IC.Backligh
tCompensationAuto);
```

```
    end
    else
    begin

InitSliderAndCheckbox(8,IC.BacklightCompensationAvailable,IC.BacklightCompen
sationRange,IC.BacklightCompensation,False,False);
    end;
  end
  else
  begin
    InitSliderAndCheckbox(8, False, 0, 0, False, False);
  end;

  //Ganancia
  if IC.GainAvailable then
  begin
    if IC.GainAutoAvailable Then
    begin
      InitSliderAndCheckbox(9,True,IC.GainRange,IC.Gain,True,IC.GainAuto);
    end
    else
    begin

InitSliderAndCheckbox(9,IC.GainAvailable,IC.GainRange,IC.Gain,False,False);
    end;
  end
  else
  begin
    InitSliderAndCheckbox(9, False, 0, 0, False, False);
  end;
```

```
//PROPIEDADES DE LA CÁMARA

//Pan
if IC.PanAvailable then
begin
  if IC.PanAutoAvailable Then
  begin
    InitSliderAndCheckbox(10,True,IC.PanRange,IC.Pan,True,IC.PanAuto);
  end
  else
  begin

InitSliderAndCheckbox(10,IC.PanAvailable,IC.PanRange,IC.Pan,False,False);
  end;
end
else
begin
  InitSliderAndCheckbox(10, False, 0, 0, False, False);
end;

//Inclinación
if IC.TiltAvailable then
begin
  if IC.TiltAutoAvailable Then
  begin
    InitSliderAndCheckbox(11,True,IC.TiltRange,IC.Tilt,True,IC.TiltAuto);
  end
  else
  begin
    InitSliderAndCheckbox(11,IC.TiltAvailable,IC.TiltRange,IC.Tilt,False,False);
  end;
```

```
end
else
begin
 InitSliderAndCheckbox(11, False, 0, 0, False, False);
end;

//Enrollar
if IC.RollAvailable then
begin
 if IC.RollAutoAvailable Then
 begin
  InitSliderAndCheckbox(12,True,IC.RollRange,IC.Roll,True,IC.RollAuto);
 end
 else
 begin
  InitSliderAndCheckbox(12,IC.RollAvailable,IC.RollRange,IC.Roll,False, False)
 end;
end
else
begin
 InitSliderAndCheckbox(12, False, 0, 0, False, False);
end;

//Acercamiento
if IC.ZoomAvailable then
begin
 if IC.ZoomAutoAvailable Then
 begin
  InitSliderAndCheckbox(13,True,IC.ZoomRange,IC.Zoom,True,IC.ZoomAuto);
 end
 else
```

```
    begin

InitSliderAndCheckbox(13,IC.ZoomAvailable,IC.ZoomRange,IC.Zoom,False,False)
;
     end;
   end
   else
   begin
     InitSliderAndCheckbox(13, False, 0, 0, False, False);
   end;

   //Fotografía
   if IC.ExposureAvailable then
   begin
     if IC.ExposureAutoAvailable Then
     begin

InitSliderAndCheckbox(14,True,IC.ExposureRange,IC.Exposure,True,IC.Exposure
Auto);
      end
      else
      begin

InitSliderAndCheckbox(14,IC.ExposureAvailable,IC.ExposureRange,IC.Exposure,F
alse,False);
      end;
    end
    else
    begin
      InitSliderAndCheckbox(14, False, 0, 0, False, False);
    end;
```

```
  //Iris
  if IC.IrisAvailable then
  begin
    if IC.IrisAutoAvailable Then
    begin
      InitSliderAndCheckbox(15,True,IC.IrisRange,IC.Iris,True,IC.IrisAuto);
    end
    else
    begin
      InitSliderAndCheckbox(15,IC.IrisAvailable,IC.IrisRange,IC.Iris,False,False);
    end;
  end
  else
  begin
    InitSliderAndCheckbox(15, False, 0, 0, False, False);
  end;

  //Foco
  if IC.FocusAvailable then
  begin
    if IC.FocusAutoAvailable Then
    begin

InitSliderAndCheckbox(16,True,IC.FocusRange,IC.Focus,True,IC.FocusAuto);
    end
    else
    begin

InitSliderAndCheckbox(16,IC.FocusAvailable,IC.FocusRange,IC.Focus,False,Fals
e);
```

```
    end;
  end
  else
  begin
    InitSliderAndCheckbox(16, False, 0, 0, False, False);
  end;

  //Color disponible
  if IC.ColorEnableAvailable then
  begin
    if IC.ColorEnableAutoAvailable Then
    begin

InitSliderAndCheckbox(17,True,IC.ColorEnableRange,IC.ColorEnable,True,IC.Col
orEnableAuto);
    end
    else
    begin

InitSliderAndCheckbox(17,IC.ColorEnableAvailable,IC.ColorEnableRange,IC.Color
Enable,False,False);
    end;
  end
  else
  begin
    InitSliderAndCheckbox(17, False, 0, 0, False, False);
  end;

  //Llenar el combo con la tasa de frames soportados
  // por el dispositivo de captura
  ObtenerTasasFrameDisponibles();
```

```
  end;
end;

{*********************************************************
* InitSliderAndCheckbox
* Habilita/Deshabilita la etiqueta, barra y check box de
* una propiedad
*********************************************************}
procedure TfrmConfiguracion_imagen.InitSliderAndCheckbox(IndicePropiedad :
integer;
                        ConfiguracionDisponible : boolean;
                        RangoSlider : OleVariant;
                        ValorSlider : integer;
                        AutoDisponible : Boolean;
                        AutoValor : boolean);
var
  lbl : Tlabel;
begin
  if ConfiguracionDisponible then
  begin
    if RangoSlider[1] <= RangoSlider[0] then
    begin
      ConfiguracionDisponible := FALSE;
    end;
  end;
  SetSlider(IndicePropiedad,ConfiguracionDisponible,RangoSlider,ValorSlider);
  SetCheckbox(IndicePropiedad,AutoDisponible,AutoValor);
end;

{*********************************************************
* SetSlider
```

```
* Habilita/Deshabilita una barra y le pone rango y valor
*******************************************************}
procedure TfrmConfiguracion_imagen.SetSlider(IndicePropiedad : integer;
                    ConfiguracionDisponible : boolean;
                    RangoSlider : OleVariant;
                    ValorSlider : integer);
var
  sld : TTrackBar;
  icomponente:integer;
begin
  for icomponente:=0 to ComponentCount-1 do
  begin
    if components[icomponente] is TTrackbar then
    begin
      If (components[icomponente] as TTrackbar).Tag=IndicePropiedad then
      begin
        sld:=(components[icomponente] as TTrackbar);
      end;
    end;
  end;

  if ConfiguracionDisponible then
  begin
    if (RangoSlider[1] > RangoSlider[0]) Then
    begin
      sld.Enabled := TRUE;
      if (RangoSlider[1] > sld.Min) then
      begin
        sld.Max := RangoSlider[1];
        sld.Min := RangoSlider[0];
      end
```

```
    else
    begin
      sld.Min := RangoSlider[0];
      sld.Max := RangoSlider[1];
    end;
    sld.Position := ValorSlider
  end
  else
  begin
    sld.Enabled := FALSE;
  end;
 end
 else
   sld.Enabled := FALSE;
end;

{*********************************************************
* SetCheckbox
* Habilita/Deshabilira un check box y le pone su estado
*********************************************************}
procedure TfrmConfiguracion_imagen.SetCheckbox(IndicePropiedad : integer;
                                    AutoDisponible : boolean;
                                    AutoValor : boolean);
var
  chk : TCheckBox;
  sld : TTrackBar;
  icomponente:integer;
begin
  for icomponente:=0 to ComponentCount-1 do
  begin
    if components[icomponente] is TTrackbar then
```

```
 begin
   If (components[icomponente] as TTrackbar).Tag=IndicePropiedad then
   begin
     sld:=(components[icomponente] as TTrackbar);
   end;
 end;
end;

for icomponente:=0 to ComponentCount-1 do
begin
 if components[icomponente] is TCheckBox then
 begin
   If (components[icomponente] as TCheckBox).Tag=IndicePropiedad then
   begin
     chk:=(components[icomponente] as TCheckBox);
   end;
 end;
end;


if AutoDisponible then
begin
  chk.Enabled := TRUE;
  if AutoValor then
  begin
    chk.Checked := TRUE;
    sld.Enabled := FALSE;
  end
  else
  begin
    chk.Checked := FALSE;
```

```
    sld.Enabled := TRUE;
   end;
  end
  else
  begin
   chk.Enabled := FALSE;
  end;
end;

{*********************************************************
* Eventos de las barras y check boxes
*********************************************************}

{*********************************************************
* TrackBar1Change
* Evento que registra si las barras han sido movidas
*********************************************************}
procedure TfrmConfiguracion_imagen.TrackBar1Change(Sender: TObject);
begin
  case (Sender as TTrackBar).Tag of
    1: IC.Brightness := (Sender as TTrackBar).Position;
    2: IC.Contrast := (Sender as TTrackBar).Position;
    3: IC.Hue := (Sender as TTrackBar).Position;
    4: IC.Saturation := (Sender as TTrackBar).Position;
    5: IC.Sharpness := (Sender as TTrackBar).Position;
    6: IC.Gamma := (Sender as TTrackBar).Position;
    7: IC.WhiteBalance := (Sender as TTrackBar).Position;
    8: IC.BacklightCompensation := (Sender as TTrackBar).Position;
    9: IC.Gain := (Sender as TTrackBar).Position;
    10: IC.Pan := (Sender as TTrackBar).Position;
    11: IC.Tilt := (Sender as TTrackBar).Position;
```

```
    12: IC.Roll := (Sender as TTrackBar).Position;
    13: IC.Zoom := (Sender as TTrackBar).Position;
    14: IC.Exposure := (Sender as TTrackBar).Position;
    15: IC.Iris := (Sender as TTrackBar).Position;
    16: IC.Focus := (Sender as TTrackBar).Position;
    17: IC.ColorEnable := (Sender as TTrackBar).Position;
  End;
end;

{*********************************************************
* CheckBox1Click
* Evento que registra si un check box ha sido habilitado
*********************************************************}
procedure TfrmConfiguracion_imagen.CheckBox1Click(Sender: TObject);
begin
  case (Sender as TCheckBox).Tag of
    1: IC.BrightnessAuto:= (Sender as TCheckBox).Checked;
    2: IC.ContrastAuto := (Sender as TCheckBox).Checked;
    3: IC.HueAuto := (Sender as TCheckBox).Checked;
    4: IC.SaturationAuto := (Sender as TCheckBox).Checked;
    5: IC.SharpnessAuto := (Sender as TCheckBox).Checked;
    6: IC.GammaAuto := (Sender as TCheckBox).Checked;
    7: IC.WhiteBalanceAuto := (Sender as TCheckBox).Checked;
    8: IC.BacklightCompensationAuto := (Sender as TCheckBox).Checked;
    9: IC.GainAuto := (Sender as TCheckBox).Checked;
    10: IC.PanAuto := (Sender as TCheckBox).Checked;
    11: IC.TiltAuto := (Sender as TCheckBox).Checked;
    12: IC.RollAuto := (Sender as TCheckBox).Checked;
    13: IC.ZoomAuto := (Sender as TCheckBox).Checked;
    14: IC.ExposureAuto := (Sender as TCheckBox).Checked;
    15: IC.IrisAuto := (Sender as TCheckBox).Checked;
```

```
    16: IC.FocusAuto := (Sender as TCheckBox).Checked;
    17: IC.ColorEnableAuto := (Sender as TCheckBox).Checked;
  End;
end;

{*********************************************************
* ObtenerTasasFrameDisponibles
* Si el dispositivo tiene tasas de frames obtenerlas.
*********************************************************}
procedure TfrmConfiguracion_imagen.ObtenerTasasFrameDisponibles();
var
  ColeccionTasasFrame : DeviceFrameRates;
  i : integer;
begin
  //No olvidar limpiar el combo box
  cboTasas_Frame.Clear;

  //Obtener la colección de tasas de frame del IC
  ColeccionTasasFrame := IC.DeviceFrameRates;
  if ColeccionTasasFrame.Count > 0 Then
  begin
    HabilitaControlesTasasFrame(TRUE);
    //Obtener todas las tasas de frame y ponerlas en el combo
    for i:=1 to ColeccionTasasFrame.Count do
      cboTasas_Frame.ItemIndex := cboTasas_Frame.ItemHeight;
  end
  else
    HabilitaControlesTasasFrame(FALSE);
end;

{*********************************************************
```

```
* cboTasas_FrameClick
* Elegir un tasa de frame del combo y ponerlo como la
* tasa actual del dispositivo de captura
*********************************************************}
procedure TfrmConfiguracion_imagen.cboTasas_FrameClick(Sender: TObject);
var
  TasaSeleccionada : integer;
  DevFrameRateCol : DeviceFrameRates;
begin
  //Obtener la coleccion DevFrameRateCol del IC
  DevFrameRateCol := IC.DeviceFrameRates;

  //Obtener la tasa de frame seleccionada en el combo
  TasaSeleccionada := cboTasas_Frame.ItemIndex;

  //Poner la nueva tasa de captura
  if IC.LiveVideoRunning = FALSE then
    IC.DeviceFrameRate := DevFrameRateCol.Item[TasaSeleccionada]
  else
    HabilitaControlesTasasFrame(FALSE);
end;

{*********************************************************
* AlActualizarStartStop
* Este procedimiento es llamado por la forma principal si
* el botón de inicio o el de paro se han presionado.
* Algunas propiedades pueden ser habilitadas solamente si
* la captura de video está detenida.
*********************************************************}
procedure TfrmConfiguracion_imagen.AlActualizarStartStop();
begin
```

```
 if IC.LiveVideoRunning = FALSE then
 begin
   if IC.DeviceFrameRates.Count > 0 Then
     HabilitaControlesTasasFrame(TRUE)
   else
     HabilitaControlesTasasFrame(FALSE);
 end
 else
   HabilitaControlesTasasFrame(FALSE);
end;

{*********************************************************
* HabilitaControlesTasasFrame
* Habilita o deshabilita los controles usados por
* las tasas de captura
*********************************************************}
procedure TfrmConfiguracion_imagen.HabilitaControlesTasasFrame(Disponible :
boolean);
begin
  cboTasas_Frame.Enabled := Disponible;
  lblTasas_Frame.Enabled := Disponible;

  if IC.LiveVideoRunning = TRUE then
    lblTasas_Frame.Caption := 'La tasa de captura de frame no está disponible'
  else
    lblTasas_Frame.Caption := 'Tasas de captura';
 end;
 end.
 //Unidad de Procesamiento Digital

 unit UPDI;
```

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtDlgs, StdCtrls, ExtCtrls;

type
   TRGB32 = packed record
    B,G,R,A: Byte;
end;
  TRGB32Array=packed array[0..MaxInt div SizeOf(TRGB32)-1] of TRGB32;
  PRGB32Array = ^TRGB32Array;

  TfrmDPI = class(TForm)
    procedure Histograma(var Bitmap : TBitmap;
                         var HistCont : array of integer);
    procedure Histograma100(var HistCont : array of integer);
    procedure DibujaHist(var HistCont : array of integer);
    procedure RGB_Gris();
    procedure Sobel();
  private
    { Private declarations }
  public
    { Public declarations }
    HImagen1: TImage;
    GrisImagen1: TImage;
  end;

 var
  frmDPI: TfrmDPI;
```

```
implementation

{$R *.dfm}

{**************
 * HISTOGRAMA *
 **************}
// Para calcular el histograma de una imagen con escala de grises de 8 bit
// se revisa cada pixel y después se incrementa el elemento correspondiente
// en el arreglo cada vez que se cuenta un byte de un cierto valor de
// intensidad, por ejemplo, si un byte tiene una intensidad de102, se
// incrementará la posición 102 del arreglo cada vez que se halle el byte 102

procedure TfrmDPI.Histograma(var Bitmap : TBitmap;
                             var HistCont : array of integer);
var
  x, y, i, j : integer;
  pb : pbytearray;
begin
    for i := 0 to 255 do HistCont[i] := 0;
    for y := 0 to Bitmap.height - 1 do
    begin
      pb := Bitmap.scanline[y];
      for x := 0 to Bitmap.width - 1 do
      begin
        // Se deshace la referencia al apuntador para extraer el
        // byte en el bitmap
        j := pb[x];
        // Después se incrementa el conteo de pixeles en el histograma
        // para esta intensidad usando j como el índice del arreglo
```

```
          inc(HistCont[j]);
        end;
    end;
end;

// Un histograma puede tener miles de puntos de la misma intensidad.
// Para poder mostrar el histograma en un bitmap, los puntos se
// interpretarán como un porcentaje del valor máximo, de tal manera
// que los puntos se mostrarán como valores entre 0 y 100

procedure TfrmDPI.Histograma100(var HistCont : array of integer);
var maxi, i : integer;
begin
    maxi := 0;
    for i := 0 to 255 do begin    //Hallar máximo valor en arreglo
       if HistCont[i] > maxi then maxi := HistCont[i];
    end;
    If maxi < 100 then exit  //Si no, hacer cada punto del histograma <= 100
    else begin
      for i := 0 to 255 do begin
        // Expresar cada punto del histograma como un porcentaje de maxi
        // de tal manera que los puntos del histograma varien de 0 a 100
          HistCont[i] := ((HistCont[i] * 100) div maxi);
      end;
    end;
end;

// Dibujar el histograma en el bitmap

Procedure TfrmDPI.DibujaHist(var HistCont : array of integer);
var
```

```
  x : integer;
  rect : Trect;
begin
  HImagen1.picture.Bitmap.pixelformat := pf8bit;
  HImagen1.Width := 256;
  HImagen1.Height := 200;
  HImagen1.picture.Bitmap.height := 200;
  HImagen1.picture.Bitmap.width := 256;

  rect.Left := 0;
  rect.right := HImagen1.Width - 1;
  rect.Top := 0;
  rect.Bottom := HImagen1.height - 1;
  with HImagen1.Picture.Bitmap do begin
    Canvas.Brush.Color := clWhite;
    Canvas.Brush.Style := bsSolid;
    Canvas.FillRect(rect);

    Canvas.Pen.Color := clBlack;
    Canvas.Pen.Style := psSolid;
    Canvas.Pen.Width := 1;
    Canvas.MoveTo(0, HistCont[0]);

    for x := 1 to 255 do begin
      Canvas.LineTo(x, 200 - HistCont[x]);
    end;
  end;
end;

//***************************************************************************
```

```
{************************
 * RGB a nivel de Grises *
 ************************}
procedure TfrmDPI.RGB_Gris();
var
  x,y,Avg : integer;
  line : PRGB32Array;
begin
  with GrisImagen1.picture.bitmap do
  begin
    pixelformat:=pf32bit;
    width:=GrisImagen1.width;
    height:=GrisImagen1.Height;
    for y:=0 to height-1 do
    begin
      line:=scanline[y];
      for x:=0 to width-1 do
      begin
      Avg:=((line[x].R + line[x].G + line[x].B ) div 3);
      if Avg > 255 then
        Avg:=255
      else if Avg < 0 then
        Avg:=0;
      line[x].R:=Avg;
      line[x].G:=Avg;
      line[x].B:=Avg;
      end;
    end;
  end;
end;
```

```
//***************************************************************************

{*********
 * SOBEL *
 *********}

procedure TfrmDPI.Sobel();
var
  i,j:integer;
begin
  // Aplica conversión a nivel de grises
  RGB_Gris();
  // Inicia Sobel
  with GrisImagen1.Canvas.ClipRect do
  begin
    for i := Left to Right do
      for j := Top to Bottom do
        GrisImagen1.Canvas.Pixels[i,j] := GrisImagen1.Canvas.Pixels[i-1,j+1]+
                                          2*GrisImagen1.Canvas.Pixels[i,j+1]+
                                          GrisImagen1.Canvas.Pixels[i+1,j+1]-
                                          GrisImagen1.Canvas.Pixels[i-1,j-1]-
                                          2*GrisImagen1.Canvas.Pixels[i,j-1]-
                                          GrisImagen1.Canvas.Pixels[i+1,j-1];
  end;
  with GrisImagen1.Canvas.ClipRect do
  begin
    for i := Left to Right do
      for j := Top to Bottom do
        GrisImagen1.Canvas.Pixels[i,j] := GrisImagen1.Canvas.Pixels[i+1,j-1]+
                                          2*GrisImagen1.Canvas.Pixels[i+1,j]+
                                          GrisImagen1.Canvas.Pixels[i+1,j+1]-
```

```
                    GrisImagen1.Canvas.Pixels[i-1,j-1]-
                    2*GrisImagen1.Canvas.Pixels[i-1,j]-
                    GrisImagen1.Canvas.Pixels[i-11,j-1];
  end;
end;
end.
```

ANEXO B

CÁMARA DIGITAL DFK 4303

Las cámaras usadas en el campo del procesamiento digital de imágenes están formadas por dos partes fundamentales: la *unidad de adquisición de imagen* y la *unidad de salida de la imagen.* La unidad de adquisición se basa en tubos de captura o *chips CCD.* Actualmente los chips CCD dominan en el mercado de cámaras digitales. La unidad de salida genera la señal de video adecuada para los dispositivos subsecuentes de procesamiento digital de imágenes. En cámaras estándar la imagen adquirida es transformada en una señal de video que cumpla con uno de los estándares internacionales de video, ya sea CCIR(Comité Consultatif International des Radiocommunications) o EIA (Electronics Industries Association). Las unidades de salida de cámaras no estándar están sujetas a ciertas áreas de investigación. Sin embargo, desde un punto de vista económico, estos estándares son tan fuertes que ya no es posible deshacerse de ellos totalmente.

Una opción de cámaras no estándar muy importante es la llamada salida de píxel por reloj (píxel clock output). Esto significa que el reloj internacional que controla el transporte de un píxel del chip CCD a la unidad de salida es accesible para los dispositivos subsecuentes de procesamiento digital de imágenes. Tal precisión de transferencia del píxel es un prerrequisito para la metrología precisa.

Otro método común de sincronización consiste en *disparar (trigger)* la cámara por un evento externo. Esto es especialmente importante en el caso de objetos móviles, los cuales por ejemplo, disparan la cámara vía una barrera fotoeléctrica.

Contenido del paquete.

La cámara es entregada con los siguientes elementos:

- Guía de usuario.
- La cámara, incluyendo una tapa para evitar que cuerpos extraños se adhieran al chip CCD.
- Un conector Y/C para fabricar cables propios. Este conector es ideal para la transmisión de señales Y/C (S-Video).
- Un conector para la conexión de lentes de auto iris.

Accesorios para iniciar captura

Para iniciar la captura de imágenes usando la cámara digital DFK 4303, se recomiendan los siguientes accesorios:

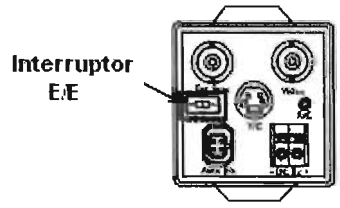
- Una unidad adecuada de suministro de energía. Imaging Source recomienda DFK 4303/Mainy. De manera alterna se puede usar una unidad de alta calidad con una salida de 12VDC/500mA.
- Un cable para conectar la cámara a un dispositivo de salida (e.g. monitor). Una terminal del cable debe ser un conector BNC. La otra terminal depende del tipo de dispositivo de salida.
- Juego de lentes tipo CS-mount o C-mount. Imaging Source ofrece una amplia variedad de dichos lentes, sin embargo para propósitos de instalación y pruebas cualquier tipo de lentes pueden ser usados.

Tipos de conexión

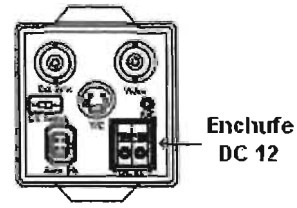
Uso de un monitor.

Para conectar la cámara a un monitor, léanse los siguientes pasos:

- Atorníllese un juego de lentes tipo CS o C. Para propósitos de pruebas o instalación se puede hacer uso de cualquier tipo de lentes. Si se desea usar un juego de lentes topo C-mount, se deberá atornillar, de manera adicional, un anillo espaciador de 5mm a la cámara.
- Usando el enchufe BNC, conectar la cámara a la entrada de video de un monitor NTSC o PAL.
- En la parte trasera de la cámara se deberá mover el interruptor **E/E** hacia **ON**. Nótese que si se desea usar lentes de auto iris este interruptor deberá estar exclusivamente en **OFF**.



- Usando el enchufe DC 12, conectar la cámara a una unidad de suministro de energía. Poner especial atención en la polaridad, invertir la polaridad podría dañar la cámara.



Encender el monitor y la imagen en tiempo real adquirida por la cámara deberá aparecer. La imagen en el monitor probablemente esté desenfocada. Se debe usar el anillo de enfoque de la cámara para hacer la imagen precisa.

Uso de tarjeta digitalizadora (Frame grabber)

Para usar la cámara con una tarjeta digitalizadora, se deberán seguir los pasos descritos en el uso de monitor, pero con dos cambios fundamentales:

- En vez de conectar la cámara a un monitor, se deberá enchufar en una tarjeta digitalizadora previamente configurada. Se deben seguir las instrucciones de la tarjeta digitalizadora para aprender cómo instalar su software apropiadamente. Es necesario asegurarse que la tarjeta digitalizadora sea compatible con la cámara.
- Las tarjetas digitalizadoras modernas, tales como el DFG/LC1 y el DFG/LC2, son los adecuados para este tipo de cámara. Usando el DFG/LC1 simplemente se necesita de un cable Y/C (S-video) que es bien conocido en establecimientos electrónicos. Si se hace uso del DFG/LC2, ya no es necesaria la unidad de suministro de energía ya que existe una unidad interna en dicha tarjeta digitalizadora. Como tanto la cámara como la tarjeta digitalizadora tienen un enchufe redondo de 12 pines, sólo es necesario un cable para conectarlos entre si.

Conectores de la cámara

Enchufe BNC.

La cámara DFK 4303 envía su señal de salida desde el enchufe BNC. En el caso más sencillo sólo se necesita conectar la unidad de suministro de energía al enchufe DC 12 y la cámara estará lista para usar.

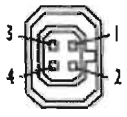
Enchufe Y/C

Haciendo uso de este tipo de enchufe (e.g. señal S-Video) se puede llevar la calidad de salida de la cámara a su máximo. El estándar Y/C fuerza que la señal de iluminación y la señal de color sean divididas a un enchufe mini DIN de 4 pines, lo cual tiene la ventaja particular de que existe una amplia gama de dispositivos de bajo costo que pueden ser conectados a la cámara. Un ejemplo de dichos dispositivos es la tarjeta digitalizadora DFG/LC1.

DFK 4303 / Y/C	
Pin	Señal
1	GND
2	GND
3	Y salida (75 Ohm)
4	C salida (75 Ohm)

Enchufe de auto iris.

El enchufe de auto iris debe usarse para conectar lentes controlados por señal de video a la cámara. Es importante poner especial atención a las instrucciones de los lentes ya que casi todos los lentes requieren diferentes cableados. Si se comete un error con las instrucciones de uso de los lentes se puede llegar a dañar el DFK 4303 y/o los lentes. Recordar que para el uso del auto iris, el interruptor E/E deberá permanecer en OFF.



DFK 4303/AI	
Pin	Señal
1	GND
2	DC salida
3	Iris Salida
4	No conectar

Configuración de la cámara

Ganancia.

El amplificador de video en la cámara DFK 4303 amplifica señales débiles de video de manera automática, de esta manera se accede al nivel máximo en la salida de video.

Obturador.

El tiempo de exposición de la cámara puede ser configurado con el obturador electrónico. Dependiendo de las condiciones locales de iluminación, este valor puede estar entre 1/50 y 1/30,000s.

Balance de blanco.

El balance de blanco ajusta la reproducción de colores de la cámara de manera automática.

Auto iris.

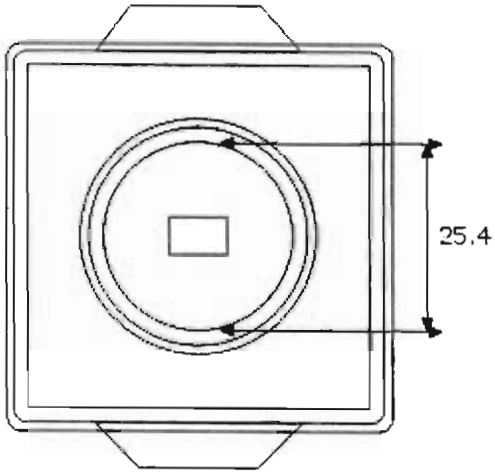
La cámara DFK 4303 puede conectarse a los lentes controlados por la señal de video. Es importante poner especial atención a las instrucciones de los lentes ya que casi todos los lentes requieren diferentes cableados. Si se comete un error con las instrucciones de uso de los lentes se puede llegar a dañar el DFK 4303 y/o los lentes. Recordar que para el uso del auto iris, el interruptor E/E deberá permanecer en OFF.

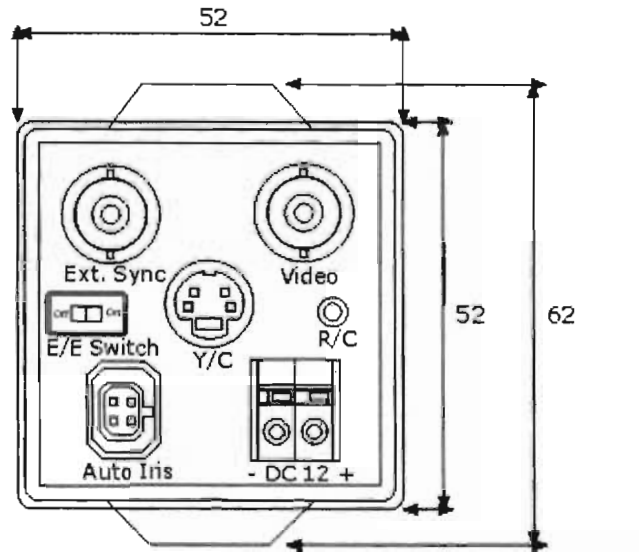
Hoja técnica

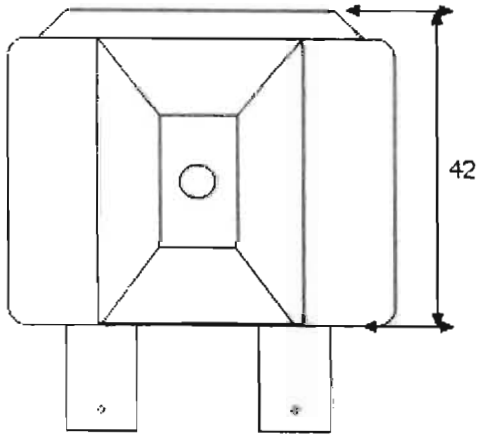
	DFK 4303
Comportamiento general (salida):	
Resolución	450 líneas
Frecuencia de fotos	PAL: 25 Frames/seg NTSC: 30 Frames/seg
Sensibilidad	0.15 lx a AGC +48dB, F1.4
Gama	0.45
SNR	> 50 db
Interface óptica	
Especificación del sensor	PAL ICX059CK, NTSC ICX058K
Tipo	Transferencia interlínea CCD
Tamaño	1/3"
Resolución	PAL: H:752, V:582 NTSC: H:768, V:494
Tamaño del pixel	CCIR: H:6.5µm, V:6.25µm, RS-170: H:6.35µm, V:7.4µm
Soporte de lentes	C/CS-mount
Interfaz eléctrica	
Voltaje de entrada	7 a 15 VDC
Consumo de corriente	aprox 200 mA (a 12 VDC)
Sincronía entrada	compuesta
Remoto entrada	RS-232
salida de video	compuesta, Y/C
salida de lentes	video
Interfaz mecánica	
Dimensiones	Alto:62 mm, Ancho:52 mm, Largo:42 mm
Masa	aprox 160g
Ajuste manual	

Obturador (remoto)	PAL: 1/50 to 1/30.000s NTSC: 1/60 to 1/30.000s
Ganancia (remoto)	0 to 48 dB
Balance blanco (remoto)	2600°C to 9000°C
Ajuste automático	
Obturador (internamente)	PAL: 1/50 to 1/30.000s NTSC: 1/60 to 1/30.000s
Obturador (remoto)	PAL: 1/50 to 1/30.000s NTSC: 1/60 to 1/30.000s
Ganancia (internamente)	0 to 48 dB
Ganancia (remoto)	0 to 48 dB
Balance blanco (internamente)	2600°C to 9000°C
Balance blanco (remoto)	2600°C to 9000°C
Medio ambiente	
Max. temperatura (operación)	-5°C to 40°C
Max. humedad (operación)	80% no condensada

Dimensiones (mm)







ANEXO C

FRAME GRABBER DFG/LC1

El DFG/LC1 es una tarjeta digitalizadora de imágenes de 32 bit con una gran calidad para aplicaciones de tipo médica o industrial a color y monocromáticas. Son soportadas las fuentes de video con salidas S-VHS (Y/C) y CVBS (video compuesto). La transmisión de datos a la PC o a la tarjeta VGA es llevada a cabo vía transferencia DMA en modo despliegue y por lo tanto provoca una carga muy baja para el CPU. Las imágenes está disponibles en calidad de color verdadero a 24 bit.

Equipo de desarrollo de software (SDKs)

Con este producto se hace la entrega de 2 quipos de desarrollo de software (SDKs) para ayudar en el uso de la tarjeta DFG/LC1. El primer equipo de desarrollo es una interfaz de aplicación (API) basada en una DLL (Dinamyc Link Library) con ejemplos de aplicaciones en C. El segundo equipo de desarrollo, FPICS, es un componente ActiveX con ejemplos de aplicación.

Los programadores en lenguaje C pueden hacer uso de la interfaz de aplicación (API) o del componente ActiveX. Para los programadores en lenguaje Visual Basic el componente ActiveX es el adecuado.

FPICS CD

Imaging CD

Contenido del paquete

- Tarjeta digitalizadora DFG/LC1 (No compatible con DEC Alpha y Apple Macintosh)
- CD-ROM con controladores para Windows 95/98/NT/2000/XP, un programa ejemplo.
- Guía del usuario.

Características

DFG/LC1

- Soporte múltiple estándar (CCIR, PAL, NTSC, RS170, SECAM).
- Soporte para todas las resoluciones VGA.
- Conversión de colores en tiempo real, de YC(YUV) a RGB.
- Soporte para color verdadero RGB 24 y RGB 32 bit.
- Resolución total de video de 768x576 a 50Hz (PAL/CCIR) y 640x480 a 60Hz (NTSC/RS170).
- Soporte para manipulación de imagen y de color de imagen.
- EEPROM en la cual se pueden escribir los datos.
- Interfaz programable para Windows (DLL).
- Captura de imágenes en formato BMP.
- Transferencia directa de datos a la tarjeta VGA y a la memoria del sistema de PC.
- Interfaz DMA con bus PCI.

Requerimientos del sistema.

- Computadora con bus PCI, que cumpla con las especificaciones PCI 2.1
- Procesador Pentium ejecutándose a un mínimo de 100 MHz (El bus PCI debe correr a 33MHz. Nótese que esto excluye a Pentium 120s y 150s.)
- Windows 95/98/NT/2000/XP
- Tarjeta PCI VGA con soporte para DirectDraw. Esto no es 100% necesario pero se obtienen mejores resultados.

Tarjetas gráficas compatibles.

Las siguientes tarjetas VGA ya han sido probadas:

- Number Nine Revolution.
- Cirrus 5446.
- S3 Trio V64+.
- S3 Trio V64V2.
- S3 Virge.
- S3 Virge DX - ELSA Victory 3DX.
- S3 Vision968.
- ET6000.
- ATI Mach64.
- Matrox Mystique.
- Matrox Millenium.
- Chips and Technologies 65550.
- Trident Pro and Vidia 9685.
- ATI 3D RagePro (AGP or PCI).
- CARDEX GX2 (AGP).

Es recomendable una tarjeta VGA con al menos 2MB en DRAM para aplicaciones monocromáticas y 4MB en DRAM para aplicaciones en color verdadero. Es importante que el controlador de DirectDraw de la tarjeta VGA esté instalado. Si no se instala el controlador apropiado, se puede ver el despliegue de algunas imágenes con el DFG/LC1 pero se mantendrán algunas restricciones debido a la frecuencia de refresco y las características de sobre posición (overlay).

BIBLIOGRAFÍA

Forecasting, structural time series models and the Kalman filter.
Harvey, Andrew C.
Cambridge University.
1996.

Kalman filtering techniques for radar traking.
Ramachandra, K. V.
M. Dekker .
2000.

Adaptive filter theory.
Haykin, Simon S.
Prentice Hall
1996.

Kalman filtering: theory and practice.
Grewal, Mohinder S.
Prentice Hall.
1993.

Adaptive digital filters.
Bellanger , Maurice.
M. Dekker.
2001.

Tratamiento digital de imágenes
Rafael C. González, Richard E. Woods.
Addison Wesley
1996

Introducción al tratamiento digital de imágenes
Jorge Lira Chávez
Instituto Politécnico Nacional
2002

Robust Real-Time Detection, Tracking, and Pose Estimation of Faces in Video Streams.
Kohsia S. Huang and Mohan M. Trivedi.
Contacto: http://cvrr.ucsd.edu/
Página WEB:
http://cvrr.ucsd.edu/publications/2004/ICPR04_Huang_Title.pdf

Integrated Approach of Multiple Face Detection for Video Surveillance.
Tae-Kyun Kim, Sung-Uk Lee, Jong-Ha Lee, Seok-Cheol Kee and Sang-Ryong Kim.
Contacto: taekyun@sait.samsung.co.kr
Página WEB:
http://www.mbari.org/aved/aved_publications/documents/Edgington_etal_demo_CVPR04_updated.pdf

Detecting and Tracking Animals in Underwater Video.
Duane R. Edgington, Danelle E. Cline, R.E. Sherlock, Dirk Walther and Christof Koch.
Contacto: {dedgington, robs}@mbari.org
Página WEB:

http://www.mbari.org/aved/aved_publications/documents/Edgington_etal_demo_CVPR04_updated.pdf

On the Tracking of Articulated and Occluded Video Object Motion.

Shiloh L. Dockstader and A. Murat Tekalp.

Contacto: http://www.ece.rochester.edu/~dockstad

Página WEB:

http://www.ece.rochester.edu/users/tekalp/papers/rti_shiloh.pdf

Integrating Graphics Into Video Image-Based Camera Tracking and Filtering.

Peter Wißkirchen, Klaus Kansy, Günther Schmitgen.

Contacto: wisskirchen@gmd.de

Página WEB:

http://www.fit.fraunhofer.de/~kansy/Publications/monaco96.pdf

Robust Segmentation and Tracking of Colored Objects in Video.

Theo Gevers.

Contacto:

Página WEB:

http://staff.science.uva.nl/~gevers/pub/GeversCSVT04.pdf

A Method for Dynamic Clustering of Data.

Arnaldo J. Abrantes and Jorge S. Marques.

Contacto: aja@cedet.isel.pt, jsm@isr.ist.utl.pt

Página WEB:

http://www.bmva.ac.uk/bmvc/1998/pdf/p038.pdf