BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

Facultad de Ciencias de la Computación

SISTEMA DE SEGUIMIENTO DE OBJETOS BAJO UN ENTORNO GNU/LINUX MÍNIMO

TESIS PROFESIONAL

QUE PARA OBTENER EL TITULO DE

INGENIERO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA

José Valeriano Aguilar Tlaseca

ASESORES

Dr. Manuel I. Martín Ortíz

M.C. José Martínez Carranza

H. PUEBLA DE ZARAGOZA ENERO DE 2009

Dedicatoria

A Margarita B. Tlaseca Alcaide, mi madre.

¡Gracias por todo “Má”!

Agradecimientos

A mi madre, a mi “Maguito”, a quien le debo todo, la persona más importante en mi vida, quién me enseñó sus valores, sus defectos y me dejó elegir mi camino y lo ha apoyado en todo momento. Gracias mamá, espero no defraudarte.

Dr. Manuel I. Martín Ortíz, excelente profesor, el mejor que he conocido, poseedor de un extraordinario conocimiento, don de líder y maestro, excelente persona, humilde, humano y ético… todo un ejemplo a seguir, a quién le agradezco por haber aceptado ser mi asesor y por orientarme en el camino. ¡Muchas gracias “Doc”!.

M.C. José Martínez Carranza, amigo de quien admiro su vasto conocimiento y habilidad de aprendizaje, características de los grandes genios, con quién realicé mi servicio social y que desde entonces me ha apoyado con su conocimiento, consejo y amistad. Gracias por ser mi asesor y por darme orientación durante el desarrollo de esta tesis.

M.C. Julio C. Pérez San Salvador, dueño de una inteligencia envidiable, amigo entrañable, capaz de hacer sentir en confianza al más apático. Gracias por haberme ayudado en las innumerables dudas que surgieron a lo largo del desarrollo de esta tesis, sin tu ayuda no lo habría logrado.

Ing. Ángel Marín George, compañero de estudios y relajo, inteligente y capaz, a quién le agradezco su valiosa e incondicional ayuda en los momentos de crisis de conocimiento que tuve durante el proceso de codificación de la aplicación de esta tesis.

Profr. Ugón Díaz Arcos, mi maestro de Taekwondo, amigo inigualable que en todo momento ha estado pendiente de mi formación académica.

A la familia Domínguez Jiménez, extraordinarias y humildes personas, gracias por su apoyo y amistad desde el primer día.

A la familia Tlaseca Nazario, especialmente a mi tía Fernanda a quien aprecio mucho y le agradezco su apoyo y cariño.

Gracias…

Resumen

El procesamiento digital de imágenes tiene múltiples aplicaciones a nivel industrial, militar, educativo, etc. y poco a poco se ha ido incorporando a nuestra vida cotidiana en aspectos tan simples como útiles, por ejemplo ahora contamos con cámaras web que mantienen en foco el rostro de la persona que las usa, cámaras fotográficas que detectan los rostros, las condiciones de iluminación y son capaces de usarlos para determinar el nivel de zoom y filtros necesarios así como el momento más indicado para captar un rostro sonriente, todo lo cual ayuda a la obtención de una mejor fotografía. Además, existen sistemas de visión más robustos que realizan tareas de seguridad, control de calidad, automatización de procesos, obtención de información cualitativa/cuantitativa, inteligencia militar, procesamiento de imágenes médicas, etc. La robustez de dichos sistemas lleva consigo el inconveniente de tener un alto costo de implantación, pues por ejemplo, solo para el subsistema de adquisición y procesamiento de las imágenes se suelen usar sistemas operativos especializados que traen incluidos sus propios entornos de desarrollo y el precio por las licencias de dichos sistemas operativos es muy alto, lo que hace poco viable su uso para sistemas pequeños, esto evidencía la necesidad de buscar alternativas más costeables. Es aquí donde el creciente "mundo del software libre y/o gratuito" nos brinda una magnífica gama de posibilidades para llevar a cabo tareas de procesamiento digital de imágenes.

Partiendo de la premisa de que la calidad no está siempre ligada al precio y de que el costo de los equipos de cómputo es cada vez más bajo, se ha desarrollado un software que tiene como objetivo realizar seguimiento de objetos, esto es, mostrar la secuencia de imágenes desde una cámara y mantener en la mira un objetivo previamente seleccionado, independientemente de la movilidad de dicho objetivo

Este trabajo muestra el desarrollo y la implementación de dicho sistema. Esta aplicación pretende ser la base de sistemas más robustos que contemplen mejoras en los algoritmos utilizados así como en la incorporación de accesorios y construir así herramientas de visión por computadora más especializadas.

Índice general

Índice de figuras

Capítulo 1

Introducción

Las imágenes digitales tienen, sin lugar a dudas, mucha importancia y se encuentran en aspectos de la vida diaria tan simples que a veces pasamos por alto su existencia.

La noción que tenemos de "imagen" es la de formas y colores visibles plasmados físicamente en un lienzo, en cambio una imagen digital podemos describirla como una imagen representada utilizando bits (unos y ceros), los cuales necesitan ser "interpretados" para poder ser captados por el ojo humano como un conjunto de puntos de colores llamado pixeles.

Existen muchas áreas del quehacer humano en donde se usan gráficos digitales, en fotografía con cámaras que detectan sonrisas; en medicina con sistemas médicos que manipulan imágenes para su mejor observación, correcto diagnóstico y tratamiento adecuado; en tácticas de guerra con aviones que disparan proyectiles de manera precisa gracias a un sistema que usa imágenes y mantiene en la mira el objetivo, en sistemas de posicionamiento global que nos indican el mejor camino a seguir usando imágenes geográficas reales, etc.

Una de las aplicaciones más importantes de las imágenes digitales son los sistemas de visión, que sirven de ojos artificiales y capturan el acontecer del medio en el que se encuentran para que se pueda extraer información relevante y usarla con un fin específico.

Los avances científicos y tecnológicos han disminuido notablemente el costo de implantación de los sistemas de visión, pues hoy se cuenta con dispositivos de captura, almacenamiento y procesamiento que además potencializan la eficiencia, lo cual es consecuencia directa de la llamada "Ley de Moore" [1] que expresa que los precios bajan a medida que las prestaciones suben. Esto trae mejoras significativas a uno de los avances más notables en la historia de la humanidad: la computadora, la cual aumenta sus capacidades a pasos agigantados y da la

posibilidad de resolver por software problemas que antes solo se solucionaban por hardware, haciendo posible tener lo que hoy se conoce como “Sistema de visión por computadora” ó “Sistema de visión artificial”, que es un conjunto de dispositivos (cámaras, motores, etc.) y programas código que interactúan a través de una computadora con el objetivo de interpretar el mundo que les rodea.

El seguimiento de objetos es una aplicación de los sistemas de visión, en los que se pretende mantener en foco un objeto (o grupo de ellos) en todo momento y sin importar a donde se mueva el objetivo. Esto implica el uso de dispositivos de captura de imágenes (cámaras) y algoritmos (software) adecuados, pues se requiere que el sistema sea capaz de responder de manera aceptable en el mínimo de tiempo para que el objeto de interés no pierda el foco, es decir, un sistema con respuesta en tiempo real.

En el presente capítulo se mencionan las razones que motivaron a la realización de un sistema de seguimiento de objetos, la metodología utilizada, los alcances del mismo, los requisitos para su funcionamiento y la descripción de los capítulos subsecuentes.

1.1 Motivación

A pesar de la disminución en el costo de fabricación y adquisición de hardware, el software hecho a la medida mantiene precios en ocasiones muy altos, tal es el caso de los sistemas operativos y herramientas de desarrollo, que con las mejoras de cada nueva versión viene el incremento en el coste por licencia de uso, y aunque para empresas grandes esto no represente un problema si lo es para su contraparte, las micro/pequeñas empresas y desarrolladores independientes, pues les augura una esperanza de vida muy corta ya que no compiten en igualdad de circunstancias con empresas de mayor capacidad de adquisición. Esto deviene en que el mercado potencial esté, hasta cierto punto, monopolizado por unos cuantos.

Afortunadamente la corriente de Free/Open Software ha tomado mucha fuerza y cada vez gana más adeptos, produciendo herramientas útiles a muchas actividades como creación musical, literaria, visual, de entretenimiento, oficina, etc. y no sólo al desarrollo de software. Una de las creaciones emblemáticas de la comunidad de software libre/gratuito es el sistema operativo GNU/Linux, que ha adquirido tal calidad, estabilidad y potencialidad que ahora grandes empresas como Sillicon Graphics, Novell, IBM y HP han decidido no sólo dar soporte a este sistema operativo, pues ahora también se pueden encontrar computadoras con GNU/Linux pre cargado, lo que disminuye notablemente el costo para el usuario final. Así como estas hay un número cada vez más grande de empresas y asociaciones que apoyan el movimiento Free/Open, esto aunado a la enorme comunidad de desarrolladores independientes a nivel mundial, la mayoría de ellos sin ánimo de lucro, da como resultado que hoy podamos hacer uso de herramientas que nos ayudan a obtener los mismos o mejores resultados que si se usara software propietario y que los sistemas operativos libres tengan una cada vez más amplia gama de hardware soportado.

En el desarrollo de sistemas de visión por computadora se requiere de un entorno estable, veloz y eficiente, máxime en empresas completamente dedicadas a este rubro o instituciones de investigación, esto se puede comprobar mirando que sistemas operativos dominan este mercado: QNX, Suse Linux Enterprise Real Time, Red Hawk Linux y en muchos casos hasta sistemas Microsoft Windows. La característica principal de dichos entornos es su velocidad, pues son sistemas que garantizan una respuesta en tiempo real a las necesidades de los sistemas que los

usan, el inconveniente es el costo por licencia de uso, y aunque algunos como QNX han decidido liberar el uso de su sistema operativo para fines no lucrativos esto no soluciona del todo el problema a empresas con bajo capital.

En este documento se pretende mostrar que un sistema operativo GNU/Linux con las herramientas de desarrollo adecuadas es una excelente alternativa a los sistemas operativos comerciales, pues representa un ahorro de recursos sin dejar de lado la eficacia para realizar las tareas.

1.2 Estrategia inicial

Para la realización de esta tesis se analizaron varias distribuciones GNU/Linux con el fin de elegir la mejor tomando en cuenta que debería ser un sistema operativo pequeño, rápido y estable (ver apéndice A)

Desde el inicio se pensó usar el lenguaje de programación C/C++ por su rapidez, experiencia de uso y extensa documentación, por lo que se tenía que realizar un análisis de entornos integrales de desarrollo (IDE's) para determinar cuál sería el más apropiado tomando como punto de partida que debía tener soporte para programar en dicho lenguaje y contar con un medio integrado para crear interfaces gráficas (ver apéndice B).

Una vez hechos los estudios anteriores, se llegó a la decisión de que el sistema a crear debería tener las siguientes características:

- La distribución a usar sería Gentoo GNU/Linux
- El entorno integral de desarrollo sería KDEvelop
- La interfaz gráfica sería creada usando la biblioteca QT
- Para la adquisición de imágenes se usaría una tarjeta capturadora (frame grabber) conectada a su vez a una cámara de video de tipo vigilancia

1.3 Cambios al enfoque inicial

Inicialmente se había pensado usar como medio de captura un frame grabber y una cámara de vigilancia, pero los costos en el mercado rebasaban el presupuesto con el que se contaba, por lo que se desistió de esta idea y se optó por usar una webcam USB, pues este medio de conexión no requiere de una tarjeta capturadora y hay cada vez más productos de donde elegir. La cámara web elegida fue una Logitech Quickcam Pro 900 (ver apéndice C).

También se había pensado usar como sistema operativo un entorno Gentoo GNU/Linux debido a que es una distribución minimalista que cuenta sólo con lo esencial y es rápida, pero a lo largo de su instalación y pruebas se decidió no usar dicha distribución por los siguientes problemas encontrados:

- Es una distribución que no es sencilla de instalar
- No es fácil de poner a punto
- Inestabilidad ocasional
- Es difícil configurar nuevos dispositivos, lo cual traería problemas al establecer la comunicación con la webcam

Por eso se optó por elegir otra distribución, que cubriera las deficiencias que ese encontraron para Gentoo, y según los resultados del análisis efectuado con anterioridad (apéndice 1) fue Ubuntu GNU/Linux la distribución elegida debido a lo siguiente:

- Es una distribución muy fácil y rápida de instalar
- Es fácil obtener un sistema acorde a nuestras necesidades.
- Al ser una distribución basada en GNU/Debian cuenta con una muy buena estabilidad
- Explota las características de dispositivos Plug & Play, por lo que es muy fácil hacer funcionar dispositivos USB
- Su popularidad la sitúa como la distribución número uno, lo que provoca que la gran mayoría del software disponible para cualquier distribución lo esté también para Ubuntu

Al ser Ubuntu una distribución con más paquetería pre instalada se decidió elegir como medio de almacenamiento del sistema un LiveDVD en vez de un liveCD, para tener espacio de guardar un sistema completamente funcional en todos los aspectos.

Finalmente, se decidió prescindir de la realización de un análisis de métodos de seguimiento de objetos y se decidió usar como bibliografía para ese respecto los documentos [3] y [4], de las cuales se sustenta la decisión de usar el algoritmo de *Correlación con Media Sustraída* como método de seguimiento.

1.4 Alcances

Para un seguimiento de objetos más eficaz se requiere un conjunto de técnicas adicionales, por ejemplo se podría dar el caso de que el objeto de interés se mueva muy rápido y el sistema pierda su ubicación, una manera de solucionarlo sería conocer de antemano las características del objeto de interés (forma, tamaño, etc.) y que cuando se perdiera la ubicación del objetivo se hiciera detección de movimiento en toda la imagen y así determinar regiones más pequeñas en las cuales hubo objetos móviles, y después se usaría un algoritmo de discriminación que tome en cuenta las características previamente conocidas del objeto a buscar para obtener el objeto más parecido a lo que se está buscando y tomarlo como el mismo objetivo pero en diferente posición dentro de la región visible por la cámara. Este trabajo se limita al seguimiento de un objeto seleccionado por el usuario mediante un cuadro envolvente, dejando de lado situaciones como la anteriormente mencionada.

El algoritmo de correlación utilizado funciona realizando la búsqueda de una zona que enmarca al objeto de interés (llamada región patrón) dentro de una región proporcionalmente más grande llamada ventana de búsqueda. Lo ideal sería que la región patrón se buscara en la imagen completa pero esto es muy costoso computacionalmente hablando, pues tomaría demasiado tiempo recorrer toda la imagen en busca de una zona que contenga el objeto más parecido o igual al que se está buscando. Es por eso que este trabajo se encuentra limitado a realizar seguimiento de objetos que no tengan movimientos demasiado rápidos pues de ser así la aplicación perdería la posición del objetivo, obligando al usuario a volver a seleccionarlo manualmente.

También se debe tomar en cuenta la iluminación del área en donde se va a realizar el seguimiento, pues un entorno muy oscuro o demasiado iluminado provocará que las regiones sean muy parecidas y por consiguiente el seguimiento sería muy defectuoso.

Otra limitante son las características con que cuente la computadora a utilizar, pues aunque debería funcionar en la mayoría de los equipos esto estará en función de que el hardware de la computadora esté soportado por la versión del sistema operativo utilizado, ya que al ser un liveDVD no podrán hacerse cambios persistentes a menos que se instale el sistema en disco duro.

Se debe mencionar también que la aplicación desarrollada tiene la limitación de usar una cámara fija, con la cual se tiene un rango de visión reducido respecto a la utilización de una cámara con libertad de movimiento horizontal y/o vertical, de las cuales ya existen webcams pero su uso necesita otro tipo de tratamiento de las imágenes y la información del entorno, ya que no es lo mismo realizar seguimiento de objetos con una cámara móvil que con una cámara que se mantiene fija.

Si tomamos en cuenta que tener una cámara con capacidad de movimiento y mira fija es una situación análoga a una cámara fija con una mira móvil podemos afrontar la limitante de movimiento en la cámara, esto es, en vez de movimiento físico en la cámara (para el cual se mantendría el objeto de interés en el centro de la imagen visible) se tiene una cámara fija que mantiene ubicado el objetivo en un recuadro, siempre y cuando este se encuentre dentro del rango de visión de la cámara.

Finalmente se debe dejar en claro que el sistema tendrá la capacidad de usar cualquier cámara USB como dispositivo capturador, siempre y cuando este cumpla con las especificaciones de el estándar UVC (del inglés *USB Video Class*), el cual tiene cada vez más aceptación.

1.5 Requisitos

Para el correcto funcionamiento de la aplicación desarrollada es necesario contar con un equipo de cómputo que cumpla con las siguientes características:

- Procesador basado en la arquitectura Intel x86 de 700MHz o superior
- Al menos 1 GB de memoria RAM para un mejor desempeño
- Un puerto USB funcional disponible para la conexión de la webcam
- Unidad de DVD-ROM para poder leer el LiveDVD que contendrá el sistema
- Monitor, teclado y mouse

1.6 Descripción de capítulos

Los capítulos siguientes de este documento se han organizado de esta manera:

- En el capítulo 2 se realiza una descripción de los conceptos fundamentales necesarios para comprender adecuadamente la esencia de esta tesis.
- En el capítulo 3 se deja en claro el problema a tratar, los objetivos a alcanzar y se establece una estrategia de solución. Por último, se hace un análisis de ingeniería de software utilizando diagramas de casos (UML) para facilitar la implantación del sistema.
- En el capítulo 4 se presenta un conjunto de diagramas de secuencia UML correspondientes a los procesos más importantes del sistema de seguimiento. Se describe el funcionamiento general de la herramienta que hizo posible la comunicación con la webcam y finalmente se menciona el proceso de creación del liveDVD.
- En el capítulo 5 se exponen los resultados obtenidos del sistema desarrollado, según los objetivos planteados.
- Finalmente se presentan las conclusiones obtenidas de la realización de esta tesis y se menciona el trabajo complementario que se ha dejado para el futuro.

Adicionalmente, este trabajo cuenta con 4 apéndices en donde se presentan:

- Un análisis que se hizo de varias distribuciones GNU/Linux y que fue la base para la elección de la distribución usada.
- Un estudio de los entornos integrales de desarrollo más comúnmente utilizados en GNU/Linux en donde se menciona el IDE elegido y los motivos de tal elección.
- Los detalles técnicos de la webcam usada.
- Finalmente se anexa el código fuente de las partes clave de la aplicación realizada.

Capítulo 2

Marco teórico

La intención de este capítulo es definir de manera breve y clara los conceptos relacionados a este trabajo, esto con la intensión de tener una base de conocimiento que ayude al lector a comprender adecuadamente el funcionamiento de un sistema de seguimiento.

2.1 Imagen digital

El concepto más general de imagen que tenemos es el de un objeto o conjunto de formas y colores visibles plasmados físicamente en un lienzo, pudiendo ser este: papel, roca, un bastidor, etc.

Partiendo de la idea anterior, si dividimos una imagen en pequeñas regiones (generalmente cuadrados) usando una rejilla, veremos a la imagen como una matriz o arreglo de dos dimensiones, donde cada elemento de la matriz (rejilla) es un pequeño punto (o región) al que se le denomina pixel (del inglés *picture element*), a esta representación de una imagen se le conoce como imagen digital.

La idea anterior es completamente válida y perceptible al ojo humano, sin embargo, para una computadora las imágenes digitales, como cualquier otro archivo, son un conjunto de dígitos binarios (ceros y unos) que representan la información de los pixeles de la imagen y que son interpretados mediante software para hacerlos ver como puntos de color que dan forma a los trazos en una imagen.

El tamaño de una imagen digital está en función del número de pixeles que la formen y entre mayor sea este número mayor será la calidad de la imagen, pues se reduce la incidencia de un fenómeno llamado cuantización, que muestra que utilizando pixeles no se puede representar con exactitud toda la información de un escenario. La figura 2.1 ilustra esta idea.

Figura 2.1 Imagen digitalizada de una pera. Se puede observar el fenómeno de cuantización.

Los pixeles son la unidad mínima de información de una imagen digital, cada uno de ellos representa un número cuyos valores van del 0 (negro) al 255 (blanco) y representan la intensidad luminosa en ese determinado punto o pixel. Además la calidad de la imagen digital está en función del tamaño que se le dé al pixel (BPP, de "bit por pixel"), o regresando a la analogía, de el tamaño del cuadrado de la rejilla.

Una imagen digital a colores está formada por tres capas de color, cada una de las cuales es una matriz de MxN y la superposición de las tres da como resultado que los pixeles se combinen para formar los colores de las bandas espectrales RGB (Red, Green, Blue), en cambio una imagen en escala de grises solo necesita de una capa o matriz, esto se muestra la figura 2.2

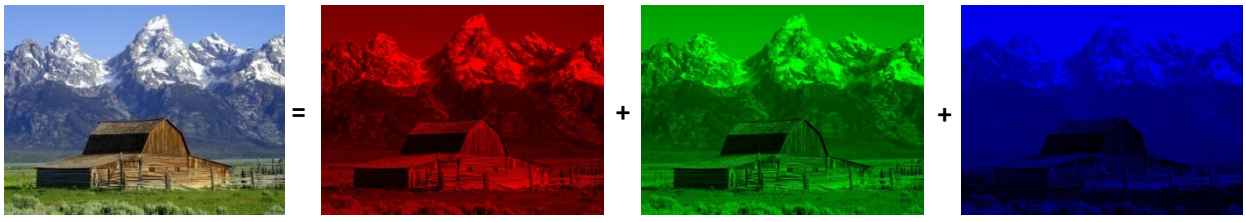


Figura 2.2 Descomposición de una imagen en sus componentes RGB

Existen varios modelos de representación del color aparte del RGB, como el HSV basado en Tono, Saturacion y Valor para cada pixel o el CMYK (Cyan, Magenta, Yellow, Black) que es análogo al RGB, pues los colores primarios en uno son los secundarios en el otro.

Formalmente, una imagen digital es una función bidimensional con dominio en los números naturales (0, 1, 2, … +∞) en donde $f(x, y)$ representa la información cromática de la imagen digitalizada.

$$f = \begin{pmatrix} f(0,0) & f(0,1) & f(0,2) & \dots & f(0,N) \\ f(1,0) & f(1,1) & f(1,2) & \dots & f(1,N) \\ f(2,0) & f(2,1) & f(2,2) & \dots & f(2,N) \\ \vdots & \vdots & \vdots & \dots & \vdots \\ f(M,0) & f(M,1) & f(M,2) & \dots & f(M,N) \end{pmatrix}$$

M y N representan el tamaño de una matriz con M número de renglones y N número de columnas.

En visión por computadora se suele tratar con imágenes en escala de grises en vez de imágenes a color, esto agiliza el tratamiento de la información. A este proceso de conversión se le conoce como promediado de color y habitualmente se hace de manera ponderada dependiendo del canal de luminosidad que se quiera realzar, la ecuación necesaria para esto es:

$$BN = alfa(R) + beta(G) + gama(B)$$

Es decir, se multiplica cada uno de los componentes del pixel por un número (alfa, beta, gamma) para determinar la luminosidad de ese canal en el modelo RGB.

Existen muchas ventajas de las imágenes digitales sobre las imágenes tradicionales, por ejemplo, si tenemos los pixeles de una imagen podemos extraer información de ellos, modificar un pixel o conjunto de ellos, aplicar filtros para mejorar la imagen, realizar operaciones aritméticas como reflexiones, rotaciones, escalamientos, etc.

2.1.1 Secuencia de imágenes

Una secuencia de imágenes no es más que un grupo de imágenes del mismo escenario tomadas en diferente instante de tiempo, con el objetivo de captar el movimiento de las formas

que componen la imagen. Habitualmente se pretende que el tiempo entre la captura de una imagen y la captura de la siguiente sea muy pequeño.

Si observamos las imágenes de una secuencia de manera ordenada y rápida se producirá la sensación de que los elementos que las forman adquieren movimiento, este es el fenómeno óptico que sustenta lo que conocemos como video y animación.

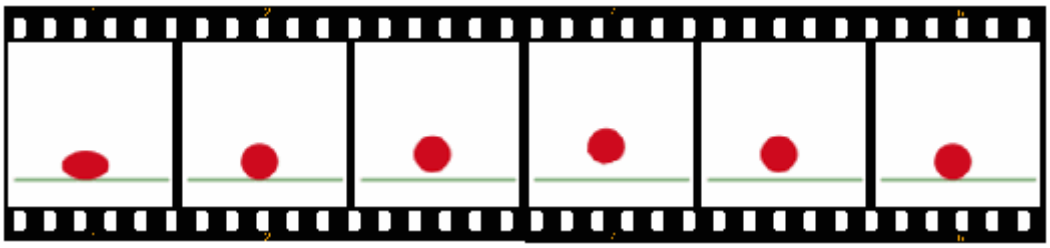


Figura 2.3 Secuencia de imágenes que al ser mostradas rápidamente darían la sensación de una pelota roja que salta

Conforme más pequeño sea el intervalo de tiempo entre captura y captura, más detalle de movimiento será el que se pueda captar y mostrar. Así pues, si se capturan 1000 cuadros por segundo de un escenario se tendrá más detalle de los movimientos que ocurran en ese tiempo que si solo se capturan 100 cuadros por segundo. Cabe mencionar que el ojo humano solo puede percibir aproximadamente unos 24 cuadros por segundo (fps, del inglés “frames per second”) como máximo, por lo cual las películas y animaciones se proyectan a esta velocidad o menos.

Existen tareas complejas que se pueden resolver usando secuencias de imágenes digitales, como reconocimiento de objetos, seguimiento de objetos, detección de movimiento, etc.

2.2 Detección de movimiento

La detección de movimiento se puede definir como la capacidad de determinar el instante en que ocurre movimiento físico en un área determinada.

Desde hace ya muchos años se han ideado mecanismos para poder captar el movimiento en una región física, así por ejemplo en el siglo XIV ya existían mecanismos manuales capaces de detectar el movimiento de animales; dichos inventos estaban formados por unas redes que atrapaban al animal que moviera un hilo tensado y colocado estratégicamente en el paso de animales. La figura 2.4 muestra un ejemplo que usa el principio descrito anteriormente.

Figura 2.4 Mecanismos antiguos para la detección de movimiento. Trampa para aves

Los avances científicos y tecnológicos a lo largo del tiempo han perfeccionado estos sistemas rudimentarios y les han dotado de una eficacia sorprendente. Una de las versiones más populares de un detector de movimiento es un dispositivo electrónico que emite un haz láser continuo y se activa cuando el haz se ve interferido entre el emisor y su destino habitual (receptor) provocando que una alarma suene. Este tipo de detector de movimiento es muy usado para vigilar entradas a establecimientos comerciales, para proteger objetos valiosos y casas habitación contra ladrones. Existen variantes, por ejemplo, modelos que bombardean continuamente (o por periodos de tiempo) el ambiente con ondas infrarrojas y cuando la uniformidad de estas ondas se ve alterada

suena una alarma que indica que se ha producido movimiento en la zona de alcance del detector. La figura 2.5 muestra este tipo de sistemas.

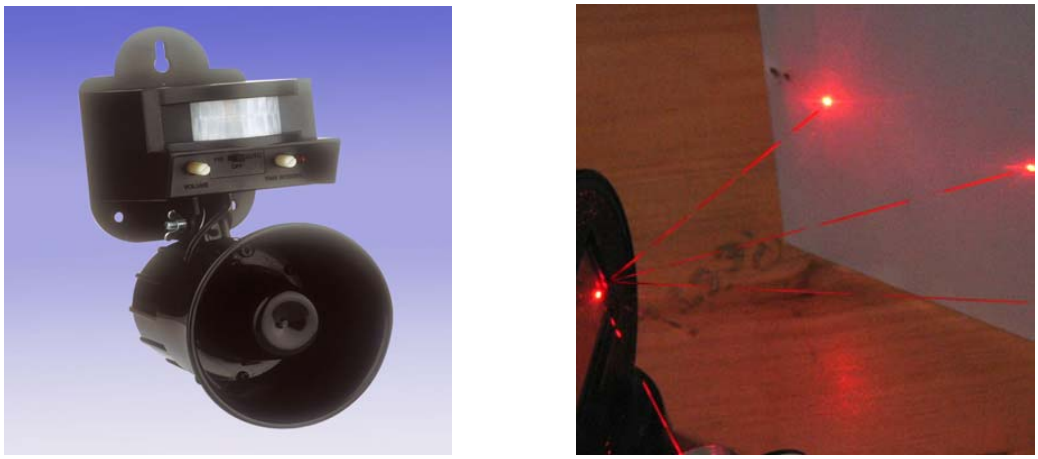


Figura 2.5 Detectores de movimiento modernos: infrarrojo y laser

En los sistemas de visión no sólo se requiere saber cuando ocurre movimiento en una área determinada, también es necesario obtener información del objeto en movimiento, tal como forma, tamaño, nueva posición dentro del área visible, etc. es aquí donde se requiere el uso de cámaras que enfoquen el área de interés y entra en juego el concepto de imagen digital, ya que para saber si en una zona ocurrió movimiento de objetos o no y la dirección del mismo se requiere de algoritmos que involucran el análisis de la información de los pixeles, lo cual ayuda a determinar las zonas de la imagen en donde hubo movimiento. Para este proceso se necesitan por lo menos dos imágenes de la misma zona de interés tomadas en diferente instante de tiempo a las cuales se les aplican.

2.2.1 Técnicas para la detección de movimiento

Existen dos tipos de movimiento a considerar, el movimiento de un plano para un objeto estático, o el movimiento del objeto sobre el plano. Sin embargo como el movimiento es finalmente relativo, ambos tipos de movimiento podrían ser considerados iguales, aunque no en

todos los casos es así, pues si se consideran efectos de iluminación y sombras el movimiento deja de ser relativo.

Se han ideado varios métodos para detectar el movimiento de objetos en el espacio y han sido clasificados de distintas maneras, algunas de las comúnmente encontradas en la literatura son las siguientes:

2.2.1.1 Diferenciación de imágenes

Es el algoritmo más sencillo para la detección de movimiento el cual busca encontrar diferencias entre imágenes sucesivas. Se realiza una comparación de la imagen actual de una secuencia con una imagen referencia, normalmente una inmediata anterior. Dicha comparación consiste en restar a los valores de cada pixel del cuadro actual los valores de cada pixel del cuadro anterior, si las imágenes comparadas son idénticas el resultado de la diferencia será nulo y en caso contrario se detectará el movimiento a través del valor obtenido en el resultado como se puede apreciar en las figuras 2.6 y 2.7

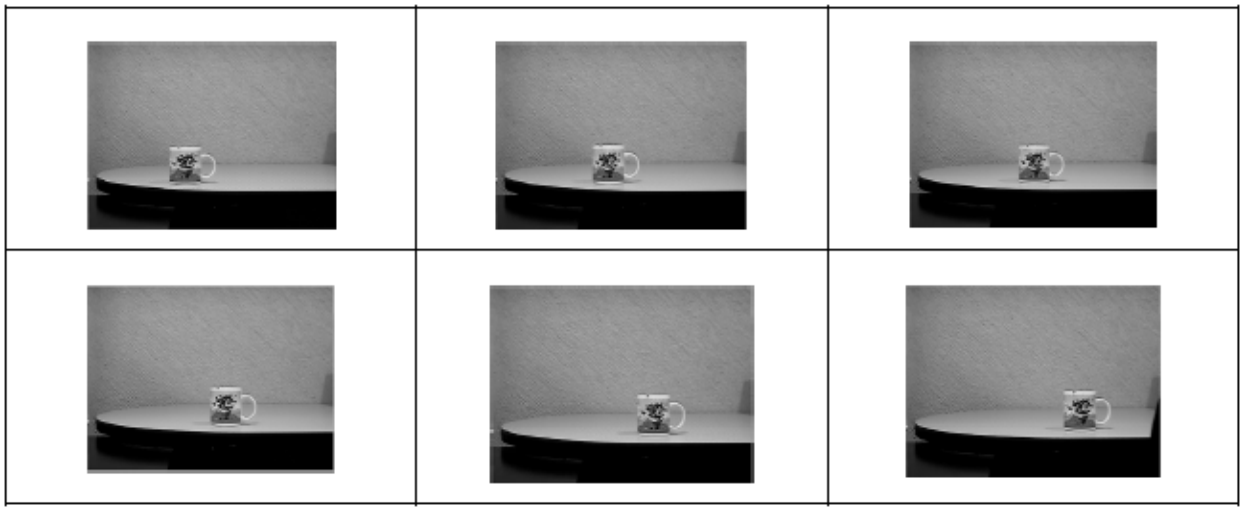


Figura 2.6 Método de diferenciación de imágenes: secuencia inicial

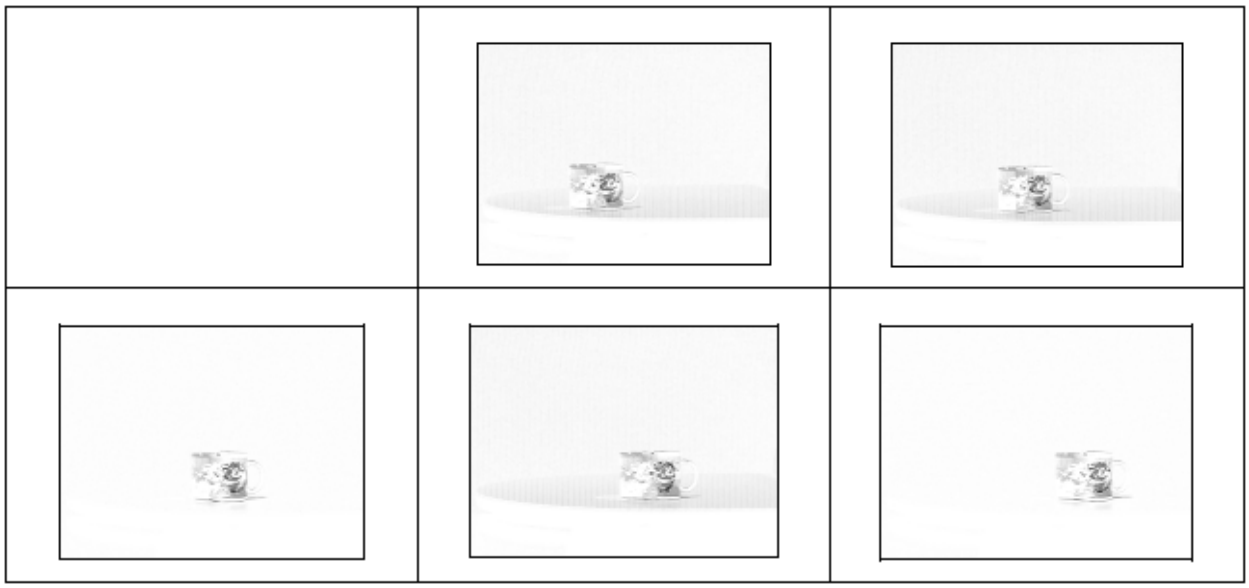


Figura 2.7 Método de diferenciación de imágenes: resultado de la diferencia

Este algoritmo no es útil cuando se busca rastrear áreas en movimiento pues prácticamente se limita a detectar cambios en las imágenes sin ser muy exacto en los resultados, además tiene otras limitaciones, pues se ve afectado si la cámara que captura las imágenes no se encuentra fija o si se producen otro tipo de cambios, por ejemplo si la iluminación del área de interés aumenta o disminuye podría dar como resultado un "falso" movimiento. Algunas aplicaciones comunes de este método incluyen sistemas de detección de intrusos, sistemas de supervisión de vehículos, sistemas de análisis de imágenes satelitales, sistemas que miden la erosión terrestre, la deforestación y el crecimiento urbano así como en análisis médicos para calcular la distribución celular.

2.2.1.2 Substracción del fondo

Su funcionamiento se basa en identificar los objetos móviles de una secuencia de imágenes, esto se hace calculando la diferencia entre una imagen de la serie con una imagen del fondo previamente obtenida, esto se muestra en la imagen 2.8

Figura 2.8 Método de sustracción del fondo para la detección de movimiento. Si contamos de antemano con el fondo de una secuencia de imágenes en las que hay objetos móviles podemos aislar los objetos de interés

Al utilizar este método se complica un poco la detección de movimiento ya que se pueden obtener errores recurrentes si no se tiene una imagen del fondo preestablecida de manera adecuada. A esto se le suma la dificultad que presenta el proceso de diferenciación de los objetos móviles que realmente interesan de aquellos que son irrelevantes, lo que requiere algoritmos muy robustos para su implementación en sistemas reales.

2.2.1.3 Estimación por movimiento de bloques

Algoritmo utilizado en la estimación de movimiento, consistente en la eliminación de redundancia temporal entre dos o más fotogramas sucesivos. Cada una de las imágenes pertenecientes a una secuencia se divide en bloques rectangulares (generalmente cuadrados) denominados macrobloques. El método pretende detectar el movimiento entre imágenes con respecto a los macrobloques que las constituyen.

Los bloques del fotograma actual son cotejados con los bloques del fotograma de destino o de referencia (anterior al actual, generalmente el primero), deslizando el actual a lo largo de una región concreta de píxeles del fotograma de destino.

Un criterio de semejanza determina la elección del bloque con mayor similitud (o que minimiza un error medido) de entre los candidatos dentro de la ventana de búsqueda de tamaño fijo del fotograma de referencia. Si el bloque elegido no se encuentra en la misma posición en ambas

frames, significa que se ha movido. La distancia del bloque coincidente entre el fotograma actual y el de referencia se define como el vector de desplazamiento estimado, y será el que se le asigne a todos los píxeles del macrobloque.

En el caso ideal, los píxeles correspondientes de los bloques coincidentes serían exactamente iguales. No obstante, ese caso sucede en muy raras ocasiones, ya que la forma de los objetos en movimiento varía con respecto al punto de vista del observador o la luz reflejada sobre su superficie, y siempre nos veremos afectados por el ruido. A continuación se presenta una figura que ilustra lo anterior.

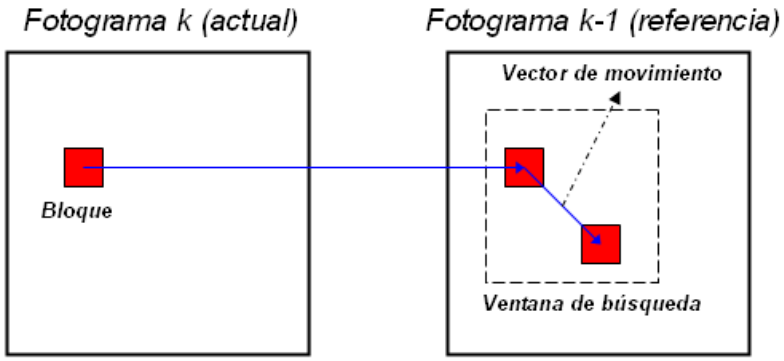


Figura 2.9 Detección de movimiento utilizando bloques

Los bloques que presentan un mismo patrón de desplazamiento pueden combinarse formando objetos en movimiento, lo cual puede resultar muy atractivo en aplicaciones de rastreo.

2.2.1.4 Flujo óptico

El flujo óptico mide el movimiento de los objetos en la imagen. Los objetos cercanos generan mucho flujo, mientras que los lejanos apenas se mueven en la retina. Muchos animales emplean flujo óptico para navegar por su entorno de modo seguro. Por ejemplo, las moscas utilizan una técnica similar para controlar su vuelo, y giran hacia izquierda o derecha tratando de igualar el flujo óptico medido en sus ojos izquierdo y derecho. De esta manera se alejan de los objetos

cercanos y se mantienen centradas en un pasillo, que es la posición en la que el flujo en ambos lados se iguala.

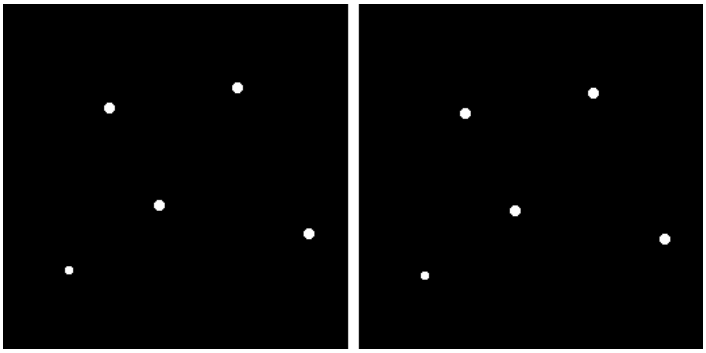
Las técnicas de detección de movimiento que usan el flujo óptico modelan el movimiento de los objetos que aparecen en la escena, o bien, la idea inversa que consiste en modelar el fondo del escenario que permanece estático para detectar posteriormente, a los objetos que se desplacen.

Dada una secuencia de imágenes, se define el flujo óptico como el campo vectorial que describe cómo varía la imagen con el tiempo, es decir, que para cada pixel de la imagen se tiene un vector de velocidad $V = (u, v)$ el cual nos indica la velocidad a la que se mueve cada pixel y la dirección de su movimiento.

Existen distintos métodos para la obtención del flujo óptico, las cuales no se profundizarán en este documento, pero a *grosso modo* se pueden encuadrar en tres grupos principales:

- **métodos basados en características**: Se basa en encontrar características de la imagen (bordes, esquinas u otras estructuras bien diferenciadas en dos dimensiones) y seguirlas según se mueven entre frame y frame de la secuencia.
- **métodos basados en gradiente**: utilizan derivadas parciales espaciales y temporales para estimar el flujo óptico en cada punto de la imagen.
- **métodos basados en correlación**: aplican medidas de correlación de fragmentos de una imagen con fragmentos de la siguiente imagen de la secuencia para determinar el flujo óptico

Un ejemplo del flujo óptico de una secuencia sencilla se muestra en la figura 2.10 (aunque no se aprecia muy bien, en la segunda imagen de la secuencia presentada los “copos de nieve” se han movido varios pixeles hacia abajo)



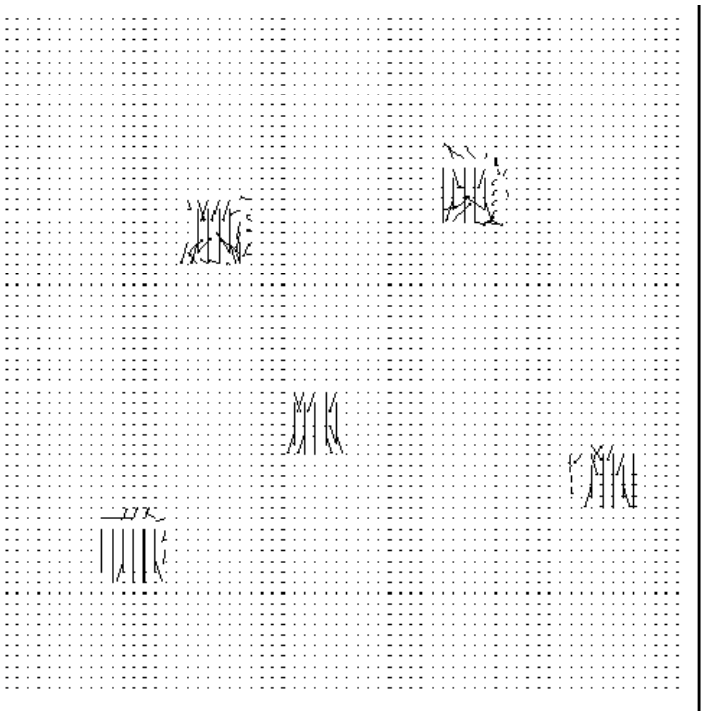


Figura 2.10 Flujo óptico del movimiento de una secuencia de puntos.

2.3 Componentes de un sistema de seguimiento

Los sistemas de seguimiento (tracking) de objetos tienen más o menos componentes dependiendo de la tarea para la que son diseñados. En la figura 2.11 se presenta un esquema generalizado de la composición de un sistema de seguimiento.

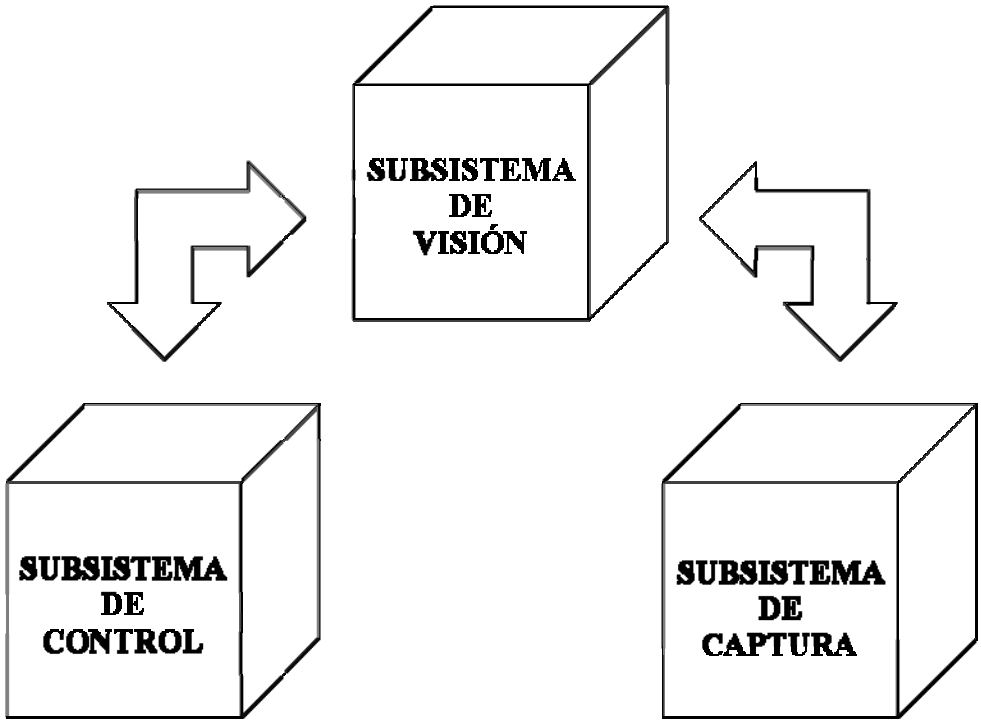


Figura 2.11 Componentes de un sistema de seguimiento

Con cada uno de estos bloques o módulos base se representan los conceptos, tareas y dispositivos que conforman el sistema, y nos da una idea de la relación que existe entre ellos. Su unión resulta ser de una gran complejidad ya que cualquier cambio en alguno de sus componentes afecta el desempeño y comportamiento del sistema en conjunto.

2.3.1 Subsistema de control

Esta parte se conforma de dispositivos de hardware como las plataformas, sobre las cuales se colocan los demás componentes físicos del sistema; los motores, que proporcionan movimiento y permiten a las cámaras capturar escenarios desde un rango amplio de movimiento; los

aditamentos que protegen las cámaras contra las inclemencias del clima y que necesitan ser controlados electrónicamente, etc.

Una parte muy importante en este componente es control de posicionamiento físico, el cual debe tener como principal característica la capacidad de no transmitir los movimientos del entorno hacia los sensores de captura; a esta propiedad se le conoce como *estabilización*.

La estabilidad de los sistemas de seguimiento es un tema esencial para su funcionamiento y su desarrollo representa un tema motivo de investigación aparte, aquí se describirá sólo brevemente. Primero se debe mencionar que los parámetros de estabilidad se miden en micro radianes y los rangos de uso se encuentran directamente ligados a dos principales causas ligadas entre sí, una es el grado de exactitud requerida por el sistema para el logro de la tarea y la otra se refiere al entorno en donde se lleva a cabo la tarea, es decir, a los movimientos externos a los que se encuentre sujeto el sistema.

Como ejemplos de sistemas de estabilización podríamos tomar: equipos de manipulación remota, de direccionamiento láser, de captura de espectros estelares, de seguimiento de objetos, etc. todos ellos cuentan con sensores y dispositivos que constituyen un sistema con el cual se mantienen invariantes ante los movimientos inerciales generados por ellos mismos, así como de los movimientos externos originados por el medio en donde se coloque el sistema. En la figura 2.12 (a) se aprecia la base de un sistema que se encuentra colocado en una embarcación marina; (b) corresponde a un sistema de vigilancia a bordo de un helicóptero y en (c) se muestran distintos tipos de sistemas auto estabilizados para aeronaves.

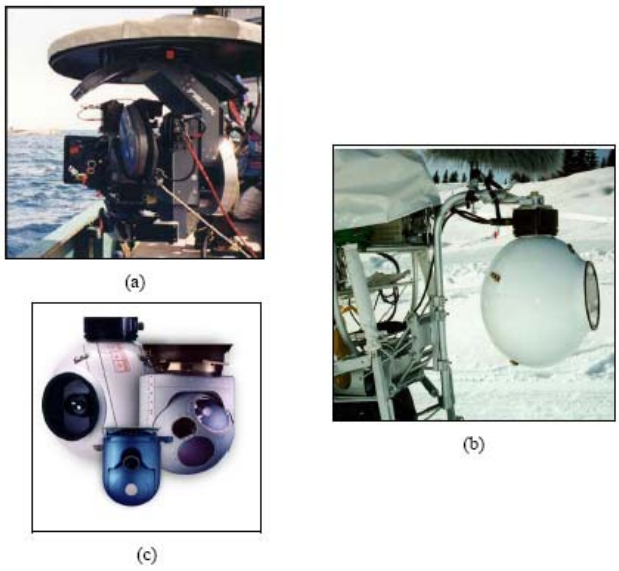


Figura 2.12 Sistemas de seguimiento de blancos móviles y de vigilancia que usan estabilizadores

Por parte del software en lo que al aspecto mecánico se refiere, se requiere el control por medio del cual se dan los comandos para efectuar las acciones de posicionamiento y compensación.
En proyectos más grandes y especializados es necesario que el sistema de posicionamiento cuente con un "sistema de control de lazo cerrado", lo que significa que el sistema tiene conocimiento de su desempeño mediante una retroalimentación propia, por lo que también a este tipo de sistemas se les conoce como "sistemas de control retroalimentado". En ocasiones este tipo de control se justifica ya que con él se obtienen las siguientes ventajas:

- La retroalimentación permite que la salida sea comparada con la entrada dada, auto ajustándose.
- Provee una exactitud que se incrementa en el tiempo, es decir, se acopla más fielmente a la posición requerida por el sistema.
- Reduce la sensibilidad a las variaciones en las características del sistema.
- Reduce el efecto de distorsión y no linealidades.

2.3.2 Subsistema de captura

Compuesto por los sensores, es decir, los dispositivos de captura de imágenes, que son la cámara y lente. La calidad y prestaciones del subsistema de captura están en función del uso que se le va a dar al sistema de seguimiento, pero por lo que respecta a la cámara, esta debe cumplir con al menos la captura de 30 fps. Por parte de la óptica empleada, es importante la capacidad del lente usado, entendiéndose por capacidad al abarque o ángulo cubierto por la lente. El abarque puede variar desde un gran angular en un extremo hasta un potente zoom en el otro. En la figura 2.13 podemos apreciar lo descrito anteriormente.

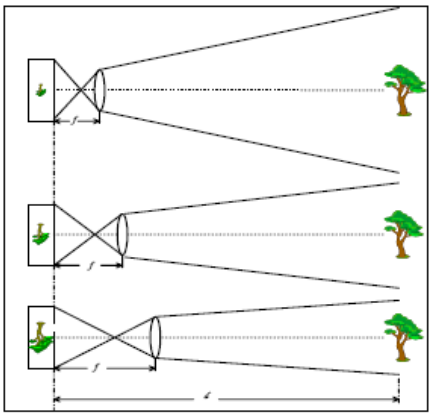


Figura 2.13 Abarque de la distancia focal

Es importante saber el abarque dado por el arreglo óptico utilizado, ya que a partir de él se conoce la resolución y máxima definición. Por medio del arreglo óptico se obtiene la representación del objeto en la imagen y se puede determinar el tamaño mínimo de los objetos en la imagen que pueden ser seguidos por el sistema.

En el caso de usar un dispositivo de distancia focal variable, es necesario que cuente con un control de posición de la lente, para que a partir de dicha posición se estime el campo cubierto por la lente, que es representado por la imagen y del cual depende la magnitud de los incrementos angulares dados al sistema de posicionamiento.

Es importante que los dispositivos de captura cuenten con mecanismos de control de algunas funciones como el zoom, que nos ayuda a acercar o alejar el objetivo a través de la lente; el foco, que sirve para ajustar la nitidez del objetivo respecto a su entorno; o el nivel de exposición, que regula el nivel de luz que es capturado por la cámara y que es útil para mejorar la calidad de las capturas en ambientes demasiado oscuros.

2.3.3 Subsistema de visión

Dentro de los aspectos de visión se consideran todos los procesos que surgen del análisis y tratamiento de la imagen. A este análisis se le conoce como *seguimiento de objetos (tracking)* y será tratado más ampliamente en el siguiente capítulo.

2.4 Seguimiento de objetos (*Tracking*)

Una aplicación común del análisis de múltiples imágenes es el seguimiento o *tracking* (por su significado en inglés) de objetos, el cual consiste en determinar la posición y velocidad de un punto (objeto o región) en una imagen dada su posición y velocidad en una secuencia anterior de imágenes.

El problema más importante que ocurre a la hora de realizar seguimiento es asociar un objeto con su representación en dos imágenes consecutivas, y para solucionarlo se han creado multitud de algoritmos.

Algunos autores diferencian dos grandes familias de métodos usados para el seguimiento del objeto de interés, las cuales son:

- Métodos basados en características.
- Métodos basados en área

Otros autores, sin embargo, hacen la siguiente clasificación:

- Seguimiento basado en marcadores
- Seguimiento basado en características naturales
- Seguimiento basado en filtros

En este trabajo se describe la segunda clasificación de las anteriormente mencionadas, sin distinción alguna, pues el objetivo no es realizar una nueva clasificación o indicar cuál es la más acertada, sino dar al lector una idea del estado del arte en el campo del seguimiento de objetos.

Dicho lo anterior se describen dichos métodos para su uso con una sola cámara, ya que los métodos de seguimiento usados en visión estéreo (dos cámaras enfocando el mismo objetivo desde distintos ángulos) generalmente combinan métodos monoculares con un tratamiento de la información de profundidad y escala, dando lugar a resultados más robustos y precisos.

2.4.1 Seguimiento basado en marcadores

Este tipo de seguimiento consiste en añadir a la escena objetos que sean fáciles de extraer de la imagen, simplificando enormemente el proceso de la estimación del movimiento.

Dentro de los marcadores se pueden distinguir dos tipos:

- **Marcadores de punto**: Reciben ese nombre porque la única información que se obtiene es un punto de correspondencia entre la escena y la imagen. Han sido muy utilizados hasta la fecha debido a que permiten la creación de un método que los extraiga e identifique de manera sencilla además de ser más fiable que los métodos basados en características naturales de los objetos. La forma circular es la más usada porque presenta poca variación bajo distorsiones en la perspectiva y su centroide es un buen estimador de su posición. Los métodos empleados para su extracción suelen basarse en algún tipo de umbralización, de la que se obtienen directamente los marcadores para luego calcular la posición de sus centroides. La figura 2.14 muestra algunos marcadores de punto.

Figura 2.14 Marcadores de punto en los cuales se utilizan distintos colores o patrones

- **Marcadores de plano**: Los marcadores de plano suelen ser rectangulares, de los cuales se extrae la posición de sus cuatro esquinas (a diferencia de los circulares de los que sólo se calculaba su centro), lo que permite estimar la "pose" del objeto (posición y rotación), aspecto de gran utilidad para sistemas que trabajen en tres dimensiones.

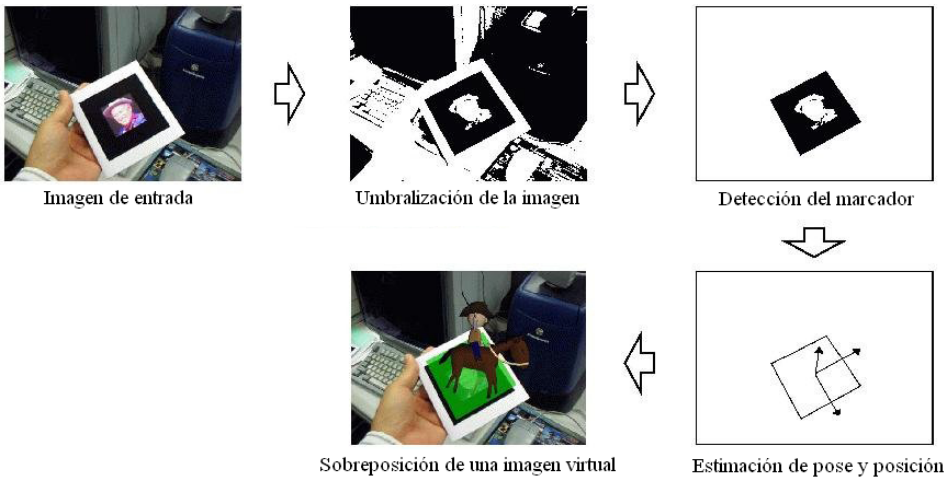


Figura 2.15 Tracking usando marcadores de plano. Una aplicación de este método es la superposición virtual

2.4.2 Seguimiento basado en características naturales

El uso de marcadores requiere modificar el entono y esto, en general, no es posible por lo que se necesitan métodos que se basen en características presentes en la imagen de manera natural.
Los métodos basados en características son apropiados para procesado de imágenes en tiempo real cuando el color y el movimiento son posibles.
Se distinguen dos grandes familias de métodos: basados en bordes y basados en la información de los pixeles interiores del objeto, como el flujo óptico.

2.4.2.1 Métodos basados en bordes

Los métodos basados en bordes se usan debido a que son tanto relativamente eficientes computacionalmente como robustas en cuanto a cambios de iluminación. Se pueden clasificar a su vez en dos aproximaciones, en la primera se buscan gradientes en la imagen sin extraer explícitamente los bordes. Los gradientes denotan una dirección en el espacio según la cual se aprecia una variación de una propiedad o magnitud física. En la segunda aproximación se extraen los bordes y se intentan acoplar a un modelo del objeto. Para cada fotograma (imagen) se extraen los bordes mientras que los segmentos del modelo se proyectan sobre la imagen de acuerdo con la predicción de su pose calculada a lo largo de toda la secuencia. Una vez hecha la proyección se emparejan los bordes, a partir de ahí se obtiene una nueva predicción de la pose que se utilizará para el seguimiento en el siguiente fotograma.

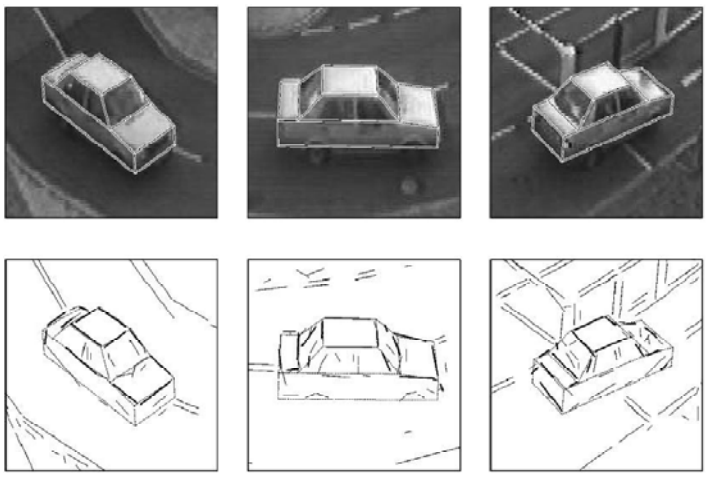


Figura 2.16 Arriba: seguimiento del coche. Abajo: Emparejamiento de la extracción de bordes con la proyección del modelo del coche

2.4.2.2 Métodos basados en la información de los pixeles

Estos métodos hacen uso de las propiedades de los pixeles que forman el objeto o región de interés, tales como el brillo, contraste, nitidez, color (nivel de gris) y vecindad. Por ejemplo, La distribución de mínimos locales de niveles de gris puede señalar la presencia de cejas, pupilas y labios y por lo tanto marcar la presencia de una cara en la imagen.

Aquí también se encuentran los métodos que usan un algoritmo de correlación, que es una función de similaridad que determina cuan parecidas son dos imágenes dada la información de los pixeles. En la literatura pueden encontrarse diversas medidas de similitud, como la *Sum of Absolute Differences*, la *Sum of Square Abosulte Differences* que es una variante de la primera. También se puede mencionar a la *Correlación Cruzada Normalizada*, a la *Correlación Basada en el Error Cuadrático Medio* o a la *Correlación con Media Sustraída.* Todas ellas son algoritmos que tienen un fundamento matemático el cual no se expondrá aquí.

En la figura 2.17 se muestra un ejemplo de seguimiento basado en color. De manera general, el proceso consiste en localizar el objeto de interés previamente definido por medio del color, en este caso una mano, dicho objeto se segmenta en un cuadrado mínimo envolvente (rectángulo interior). Posteriormente se busca la región de interés en la siguiente imagen de la secuencia en una vecindad a la región previa (rectángulo exterior) tomando como parámetro de decisión el mejor valor obtenido por una medida de similitud aplicada a toda la región de vecindad. Si se encuentra una región que se asemeje al objeto de interés se asume que el objeto se ha movido y se toma la nueva posición.

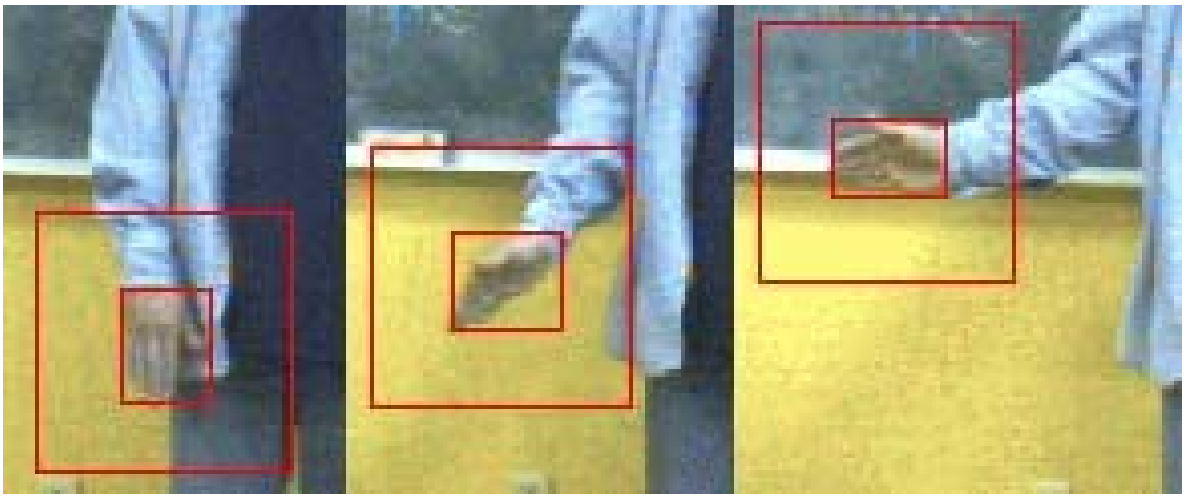


Figura 2.17 Seguimiento basado en la información de los pixeles del objeto/región de interés

2.4.3 Seguimiento basado en filtros

Un ejemplo clásico de seguimiento basado en filtros es el empleo del filtro Kalman. El filtro Kalman es un algoritmo que sirve para obtener el estado oculto de un sistema dinámico lineal. Se tiene un sistema dado por:

$$x_k = A_{k-1}x_{k-1} + B_{k-1}u_{k-1} + w_{k-1}$$
$$z_k = C_k x_k + v_k$$

Donde:

- Wk es ruido blanco de media 0 y varianza Qk en el instante k.
- Vk es ruido blanco de media 0 y varianza Rk en el instante k.

El ruido blanco es una señal aleatoria (proceso estocástico) que se caracteriza porque sus valores de señal en dos instantes de tiempo diferentes no guardan correlación estadística

El filtro Kalman permite averiguar el estado xk a partir de las mediciones anteriores de zk, uk, Qk, Rk y las estimaciones anteriores de xk. La principal característica para su uso en visión artificial es su carácter recursivo, es decir, para cada paso se basa en la información que ya ha adquirido en pasos anteriores lo que lo hace muy eficiente computacionalmente, ideal para su uso en sistemas en tiempo real. Otra de sus características es la capacidad para corregir su proceso de estimación en función de la información medida a posteriori (compara su estimación con la medición real hecha después).

Para su uso en seguimiento, la imagen se procesa y se segmenta, obteniendo el objeto a seguir. Como estado del sistema se considera la posición (x, y) del objeto en un instante k. El filtro Kalman estima la posición de objeto en la siguiente imagen, lo que restringe enormemente el espacio de búsqueda, que se centraría en un área alrededor del punto estimado, en vez de buscar por toda la imagen. Este filtro se usa mucho en aplicaciones en las que se desea tratar adecuadamente a las oclusiones, es decir, instantes de tiempo en los cuales el objeto de interés se encuentra parcial o totalmente cubierto por otro objeto que se interpone entre el objeto a seguir y la posición del observador. En las figuras 2.18 y 2.19 se muestran unas gráficas que indican el comportamiento de una aplicación que hace uso de un filtro Kalman.

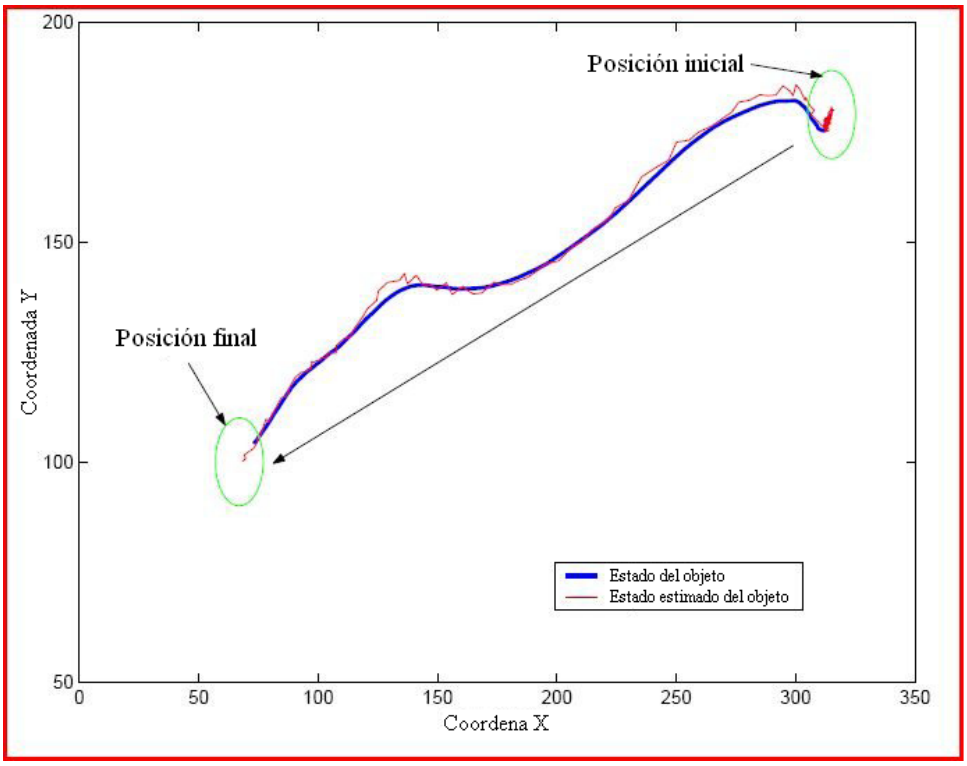


Figura 2.18 Gráfica de resultados usando un filtro Kalman para la predicción

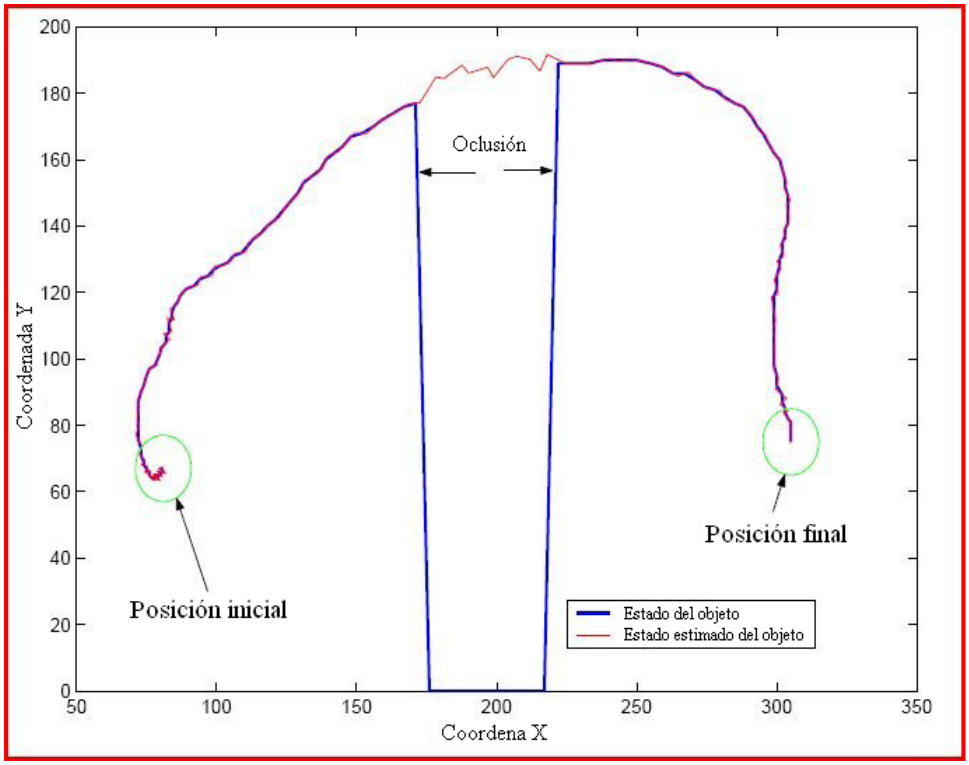


Figura 2.19 Predicción en presencia de oclusión

2.5 Aplicaciones del seguimiento de objetos

Existen en la literatura múltiples aplicaciones para el seguimiento visual mediante técnicas de visión artificial. Muchas de ellas están inspiradas en lo que nosotros, como seres humanos, hacemos con nuestro sistema de seguimiento biológico. Sin embargo, a veces resulta difícil darse cuenta de que tareas aparentemente sin nada que ver con el seguimiento, son o están compuestas en alguna medida por seguimiento visual.
El acto de leer es una de ellas, cuando nos fijamos en cada una de las palabras del texto, estamos a la vez siguiendo la línea en donde está incluida, saltando sucesivamente a las palabras situadas a la derecha de la actual.

A continuación se comentan algunas de las áreas donde un módulo de seguimiento tiene aplicación:

Navegación: De la misma manera que las personas utilizan marcas visuales para caminar o evitar objetos, un robot puede hacer uso de ellas para obtener su posición y orientación relativa a ellas durante su trayecto. Con un sistema de seguimiento multiobjetivo, es posible no sólo usar una simple marca para realizar la tarea, sino varias marcas para obtener una mejora en la fiabilidad y hacer más sencilla la obtención de la posición y orientación.

Reconocimiento de caras: El reconocimiento de caras es un problema complicado. Existen numerosos trabajos sobre este tema, los cuales tratan de identificar personas en entornos controlados y con vistas frontales de las caras. El problema se hace más difícil cuando la persona a reconocer no se encuentra inmóvil, sino que se mueve de forma natural delante de las cámaras. DESEO[4] es una plataforma de visión activa capaz de realizar seguimiento y reconocimiento de personas

Realimentación visual para brazos robot: Controlar brazos robot por medio de seguimiento visual es una técnica útil para interactuar con entornos dinámicos. Un resumen sobre control visual puede encontrarse en [5].

Vigilancia: Ésta ha sido de las primeras aplicaciones útiles para algoritmos de seguimiento. Normalmente, los sistemas de vigilancia típicos constan de cámaras estáticas con módulos de detección que activan la grabación de secuencias.

Seguimiento del movimiento de las personas: En este tipo de aplicaciones no se sigue a la persona en sí, sino a una parte de su cuerpo. Esto tiene el atractivo de que los gestos del cuerpo y las manos pueden ser reconocidos. Por ejemplo, manos y dedos [6] pueden ser usados como ratones o teclados para interactuar con un ordenador.

Control del tráfico: La monitorización de vehículos en carreteras y autopistas es otra de las aplicaciones prácticas del seguimiento. La detección de colisiones y maniobras peligrosas de los conductores pueden ser detectadas.

Armamento militar: Es aquí donde se han logrado mucho avances en el campo de los sistemas de seguimiento, si no es que la gran mayoría de ellos, pues muchas de las instituciones que realizan investigación en esta área se encuentran subsidiadas por organismos militares a los cuales prestan sus servicios ó más aún, forman parte integral de las fuerzas de defensa de una nación. Es por eso que el equipo militar de las grandes naciones cuenta con aditamentos de última tecnología. Como ejemplo claro se puede mencionar a los sistemas de navegación y seguimiento de objetivos con que están equipadas las aeronaves militares.

Capítulo 3

Planteamiento y análisis del problema

3.1 Planteamiento del problema

Toda vez que se han sentado las bases de conocimiento mínimo para el buen entendimiento de este documento, es necesario dejar en claro el problema que se pretende atacar.

Actualmente los sistemas de seguimiento están muy avanzados, utilizan técnicas de tracking y dispositivos de última tecnología, pero, ¿qué sucede con aquellos casos en los que no se dispone de mucho capital para construir un sistema dedicado exclusivamente al seguimiento de objetos, y aún más, que no se requiere un sistema tan avanzado?

Por ejemplo, si sólo se requiere un pequeño sistema que sea capaz de grabar el acontecer de una ponencia, en donde el objetivo es mantener enfocado al conferencista, pues normalmente este nunca se sale de la escena, sólo se mueve dentro de un espacio bien definido. El trabajo de un sistema de seguimiento aquí sería capturar en video al conferencista sin dejar fuera de imagen el material (proyecciones, pizarra, etc.) y así evitar el uso de una cámara manual y su correspondiente operador.

O imaginemos que se desea grabar el comportamiento de un determinado tipo de insecto, por ejemplo, la mosca de la fruta, para obtener información que ayude a entender mejor el comportamiento de dichos insectos y que ayude a resolver determinados problemas. La manera común de realizar dicha labor sería poner una cámara fija de videograbación, con lo que se obtendría una secuencia de video que sería necesario analizar posteriormente con detenimiento. En cambio, si se contara con un sistema de visión en el que se identificaran plenamente a las moscas como objetivo principal a monitorear, se podrían obtener en tiempo real datos acerca del

comportamiento de dicha especie y así se tomarían decisiones oportunamente, o bien podrían manipularse automáticamente, mediante sensores y mecanismos hardware, las condiciones de iluminación y medio ambiente para simular distintos medios naturales específicos, también en tiempo real.

Podríamos automatizar la transmisión de encuentros deportivos, pues en muchos de ellos el foco se mantiene en un objeto, por ejemplo un balón, o bien en el líder de la competencia, por ejemplo en natación. Aunque es cierto que en la élite deportiva ya se pueden encontrar aplicaciones de sistemas de seguimiento, estos tienen el inconveniente de un alto costo de implantación.

3.2 Objetivos

A continuación se describen los objetivos que se pretenden cubrir con la realización de esta tesis, los cuales se dividen en objetivo general y específicos.

3.2.1 Objetivo general

El objetivo primordial de esta tesis es demostrar que se pueden construir sistemas de seguimiento utilizando dispositivos que están al alcance de un sector cada vez más grande de la población, dispositivos como una computadora de escritorio (o laptop) y una webcam, los cuales son de uso general y cada vez más accesibles. Ahora no sólo los computólogos, grandes empresas e instituciones disponen de una computadora, pues casi cualquier persona las utiliza para sus labores habituales, desde tareas de oficina hasta cálculo matemático, entre muchas otras cosas.

Se debe mencionar que la presente tesis no tiene como objetivo construir un sistema robusto que automatice por completo actividades que involucren seguimiento de objetos como las mencionadas en el planteamiento del problema, sino dejar un sistema pequeño capaz de realizar seguimiento, y contar con la capacidad de extenderse para resolver problemas más complejos y específicos.

La característica principal de esta propuesta en su portabilidad física, pues sólo se necesitan una computadora, una webcam y lector DVD a través del cual se carga el sistema operativo GNU/Linux y la aplicación para que no sea necesario alterar el sistema operativo de la computadora utilizada. Es decir, sólo se necesitaría conectar una webcam a una PC o laptop, las cuales son fácilmente transportables, e introducir el disco de la aplicación para proceder a su utilización.

El objetivo general de la aplicación se puede resumir en el siguiente enunciado:

"A partir de una secuencia de imágenes provista por una webcam dentro de las cuales existen objetos que presentan movimiento, el sistema será capaz de mantener la ubicación del objetivo seleccionado por el usuario, siempre y cuando el objeto se encuentre dentro del rango de visión de la webcam"

3.2.2 Objetivos específicos

Ahora es necesario mencionar los objetivos específicos que se deben cubrir para obtener el sistema que se menciona en el objetivo general.

El sistema deberá contar con una aplicación de software que contemple lo siguiente:

- Estar montada en un sistema operativo GNU/Linux.
- Tener la capacidad de usar una webcam USB como dispositivo de captura.
- Contar con una interfaz simple, amigable e intuitiva que permita iniciar y detener la captura de imágenes.
- La interfaz deberá incluir controles para cambiar el tamaño del cuadro con el que se seleccionará el objetivo, y un control para aumentar/disminuir el nivel de exposición de la webcam.
- Se usará el mouse como dispositivo de selección/deselección del objetivo a seguir.
- La aplicación deberá permitir guardar una secuencia de imágenes al disco duro, para lo cual se indicará cuantas son las imágenes a capturar.
- Se usará un liveDVD como medio de almacenaje del sistema, con el cual bastará iniciar la carga del sistema operativo y la aplicación incluidos en dicho DVD para poder usar el sistema de seguimiento.

3.3 Estrategia de solución general

La parte más importante de la solución es el algoritmo de seguimiento. En este caso se ha elegido utilizar un algoritmo que hace uso de la información de los pixeles para determinar el parecido que existe entre dos imágenes. El método a usar consiste en una medida de similitud llamada *Correlación con Media Sustraída CMS* la cual ha sido utilizada en diversos trabajos con buenos resultados [2].

Todas las correlaciones o funciones de similaridad se ven afectadas por los cambios de luz o bajo contraste, pero la *CMS* se ha mostrado como una medida más robusta que, en combinación con otras técnicas, puede resultar de mayor efectividad que otro tipo de medidas [8].

La siguiente ecuación corresponde a la *CMS* cuyo rango va de [-1, 1], con -1 como la similaridad mas baja y 1 como la más alta.

$$C(a,b) = \frac{\sum_x \sum_y [T(x,y) - \overline{T}][I(x+a, y+b) - \overline{I}]}{\left\{ \sum_x \sum_y [T(x,y) - \overline{T}]^2 \sum_x \sum_y [I(x+a, y+b) - \overline{I}]^2 \right\}^{\frac{1}{2}}}$$

De la ecuación anterior se observan los elementos $\overline{T}$ e $\overline{I}$, los cuales representan el promedio o media cromática de la subimagen modelo y la subimagen del blanco candidato; esto quiere decir que a *T* e *I* corresponderían las matrices con los pixeles de la subimagen modelo y subimagen candidata respectivamente, ambas de la misma dimensión.

Si tomamos un vector *(x, y, t)* como las coordenadas de la esquina superior izquierda de la subimagen que contiene el objeto de interés en un instante de tiempo *t*, su cambio de posición en el siguiente instante de tiempo estará dado por *(x+h, y+k, t+l)*, donde *h* y *k* se espera que sean constantes de cambio lineal o de algún otro tipo que pueda ser estimado, es decir, que no tengan cambios abruptos o caóticos.

Debido a que los incrementos en *h* y *k* no son discontinuos, existe una vecindad de puntos alrededor de la posición *(x, y, t)* en donde el objeto de interés podría posicionarse en el siguiente instante de tiempo. Esto lleva a la conclusión de que no es necesario buscar en toda la imagen,

sino sobre una vecindad o ventana de trabajo. A esta estrategia se le conoce como *ventana de búsqueda* y es utilizada en la mayoría de los sistemas de seguimiento ya que su uso reduce considerablemente el espacio de trabajo y por ende de procesamiento.
Para poder usar la *CMS* se debe conocer la matriz de pixeles asociada a la imagen completa, pues dentro de esta se encontrará la matriz correspondiente a la ventana de búsqueda, y esta a su vez contendrá a las matrices de las regiones candidatas.

El algoritmo para generar las subimágenes o matrices candidatas consiste en un barrido o corrimiento sobre la ventana de búsqueda del cual se obtendrán las coordenadas *(a, b)* que corresponden a la esquina superior izquierda de la subimagen candidata *I*, la cual se comparará con el modelo *T* mediante el uso de la función de correlación entre imágenes.
En la siguiente figura se ilustra la idea anterior.

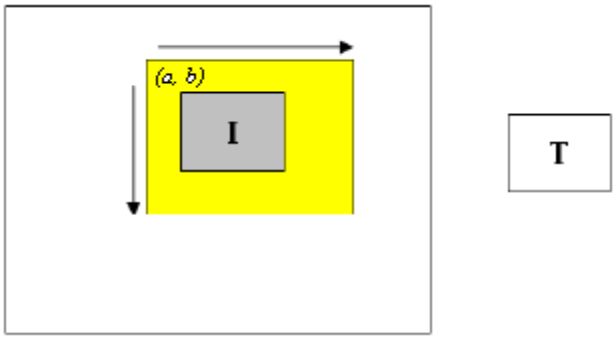


Figura 3.1 Uso de una ventana de búsqueda (recuadro amarillo) sobre la cual se aplica la medida de similaridad

Un inconveniente de la utilización de ventanas de búsqueda es determinar su tamaño, ya que no hay una regla general que sea eficaz en todos los casos. En algunos trabajos se establece acorde a la dinámica del objeto, es decir, la proporción de cambio de *h* y *k* multiplicada por un factor. En otros trabajos como en [9] se propone tomar el doble del radio de la subimagen modelo, obteniendo resultados eficientes. En este trabajo se usa la segunda propuesta; el radio de la ventana de búsqueda será el doble del radio de la ventana patrón.

Como puede apreciarse de la ecuación de la *CMS*, el tiempo de procesamiento es mayor con respecto a otras medidas de similitud, sin embargo, por su desempeño es preferible sacrificar tiempo de procesamiento por precisión en la detección.

3.4 Análisis del problema

A continuación se realiza el análisis del problema que es parte del proceso de ingeniería de software que se seguirá para la creación de la aplicación.

La imagen 3.2 muestra una figura que ilustra el modelo de desarrollo utilizado en la resolución de los objetivos de esta tesis.

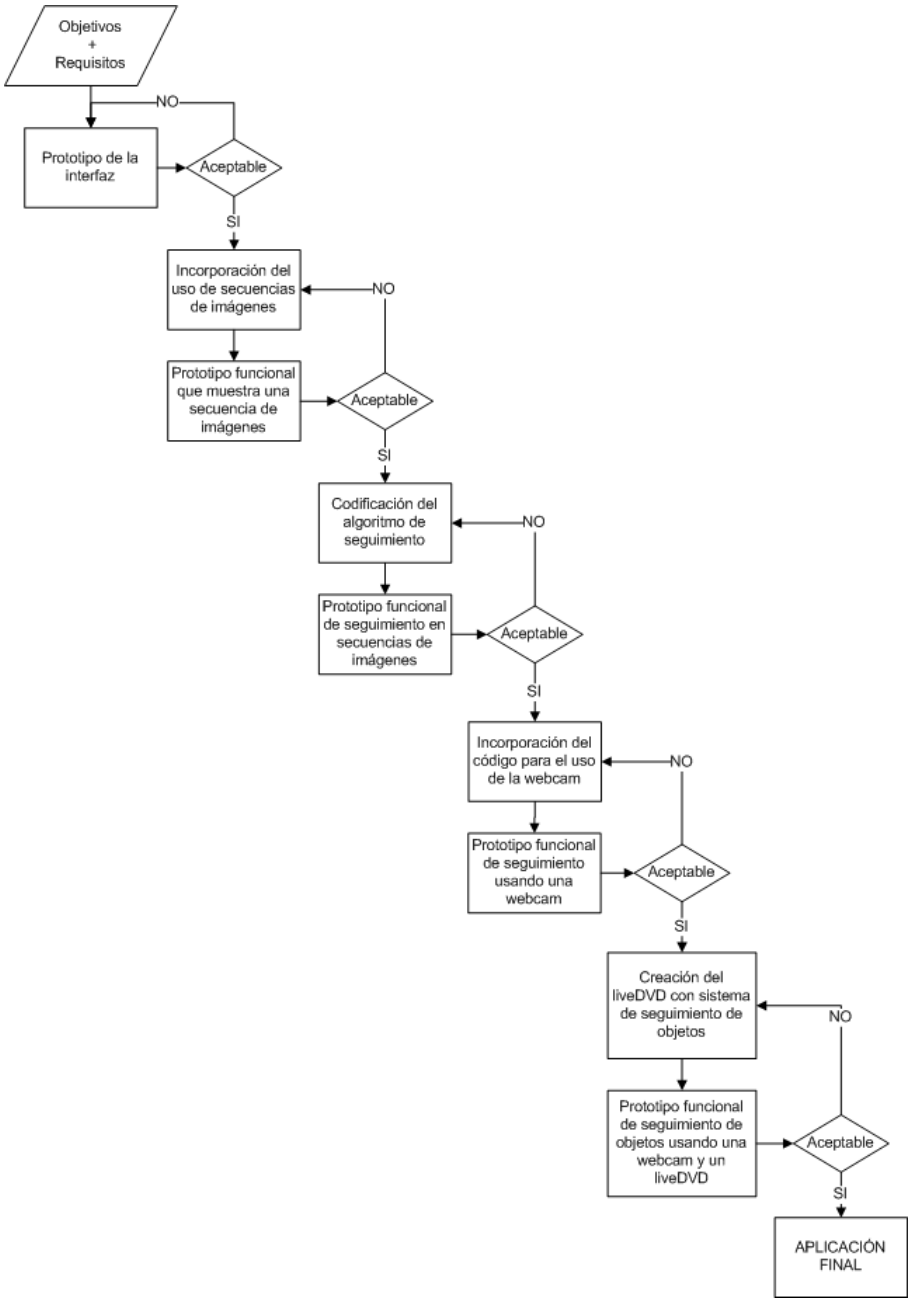


Figura 3.2 Proceso de desarrollo

A partir de aquí se usará el Lenguaje de Modelado Unificado (UML, por sus siglas en inglés *Unified Modeling Language*) para describir el análisis y diseño (capítulo 4) del sistema de seguimiento. UML se compone de varios tipos de diagramas pero aquí solo se utilizaron los que fueron útiles en el desarrollo del sistema. UML es una metodología cíclica de depuración por lo que se generan mucho diagramas; los aquí expuestos son los obtenidos en la etapa final.

Analizando las situaciones que se derivan del funcionamiento esperado del sistema se obtiene el siguiente diagrama de casos de uso:

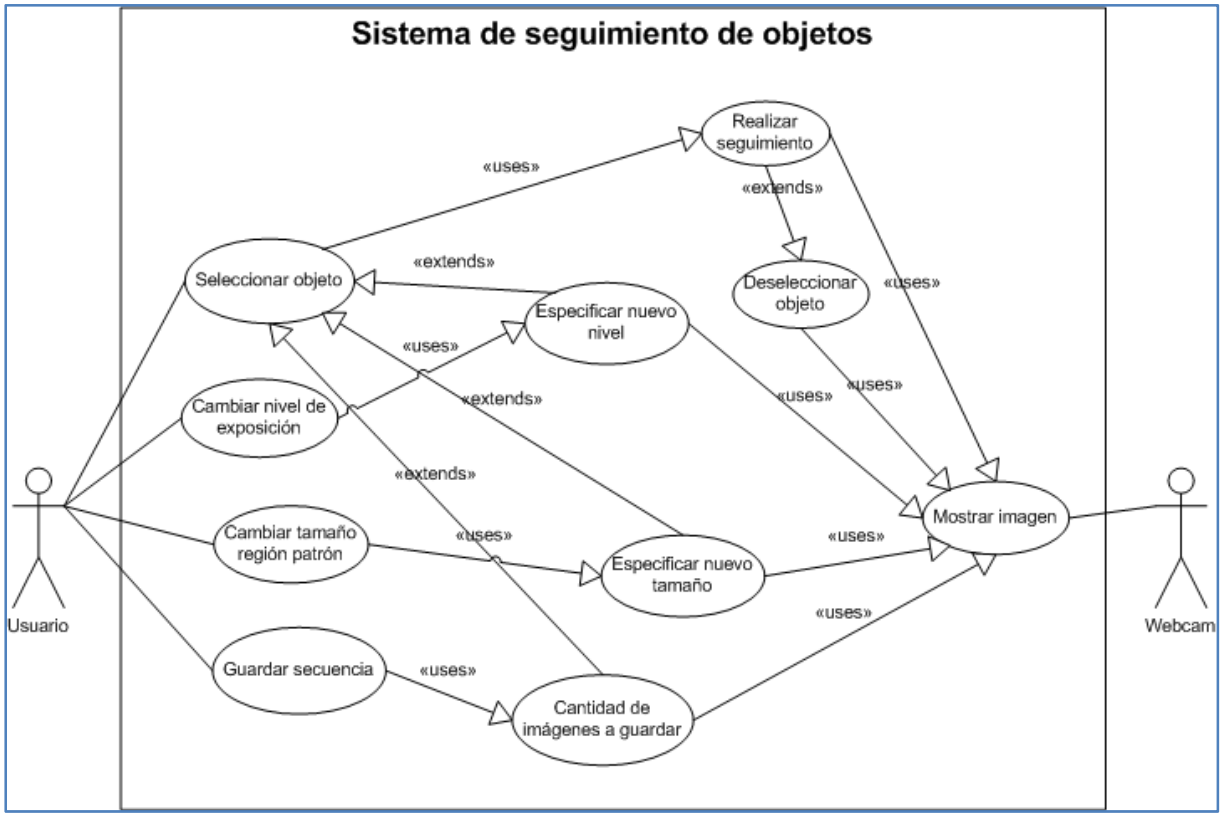


Figura 3.3 Diagrama de casos de uso del sistema de seguimiento de objetos

El diagrama de casos de uso es útil para realizar una descripción y vista generalizada de las funciones que realizan los diferentes actores del sistema. A partir del diagrama de casos de uso en la figura 3.3 se extraen los escenarios generales que se suscitan en el funcionamiento del sistema, entre los que destacan:

- El sistema captura y despliega imágenes de la webcam.
- El usuario puede seleccionar un objetivo para darle seguimiento.
- El sistema guarda secuencias con las imágenes capturadas.
- El usuario puede cambiar el tamaño de la región patrón.
- El usuario puede cambiar el nivel de exposición de la webcam.

Por cada caso de uso anterior (óvalos) se determinan y analizan las actividades que cada uno de ellos implica. Para llevarlo a cabo se emplean los diagramas de actividades; la figura 3.4 es el diagrama de actividades correspondiente a la captura cíclica de imágenes o caso de uso "Mostrar imagen".

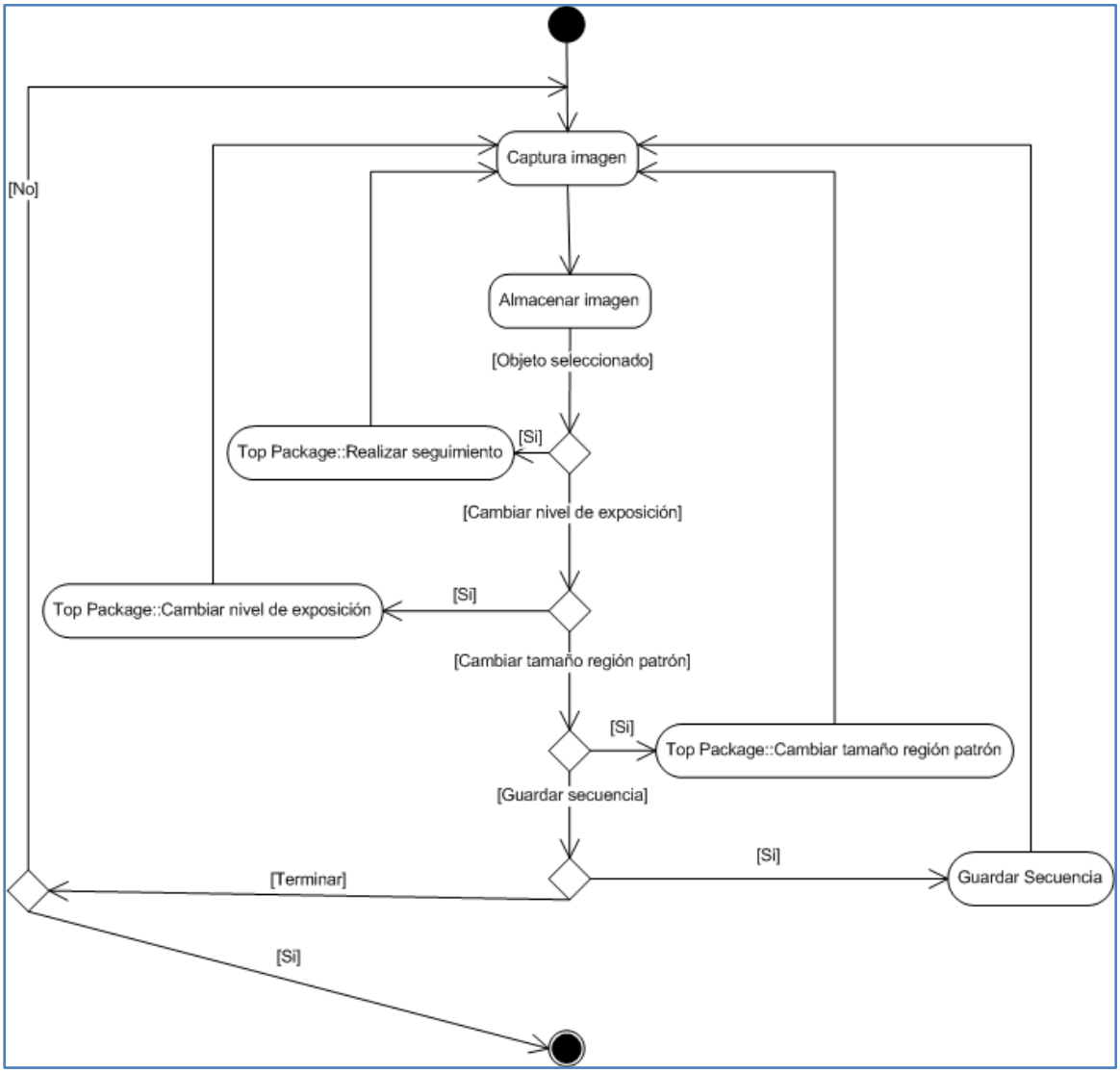


Figura 3.4 Diagrama de actividades de la captura cíclica de imágenes

En el diagrama anterior se deben realizar en el proceso cíclico de captura y despliegue de imágenes, esta última acción contenida dentro de la actividad denotada como “Almacena imagen”.

En la figura 3.5 se muestra el conjunto de tareas a llevarse a cabo dentro del caso de uso “Seleccionar objeto”.

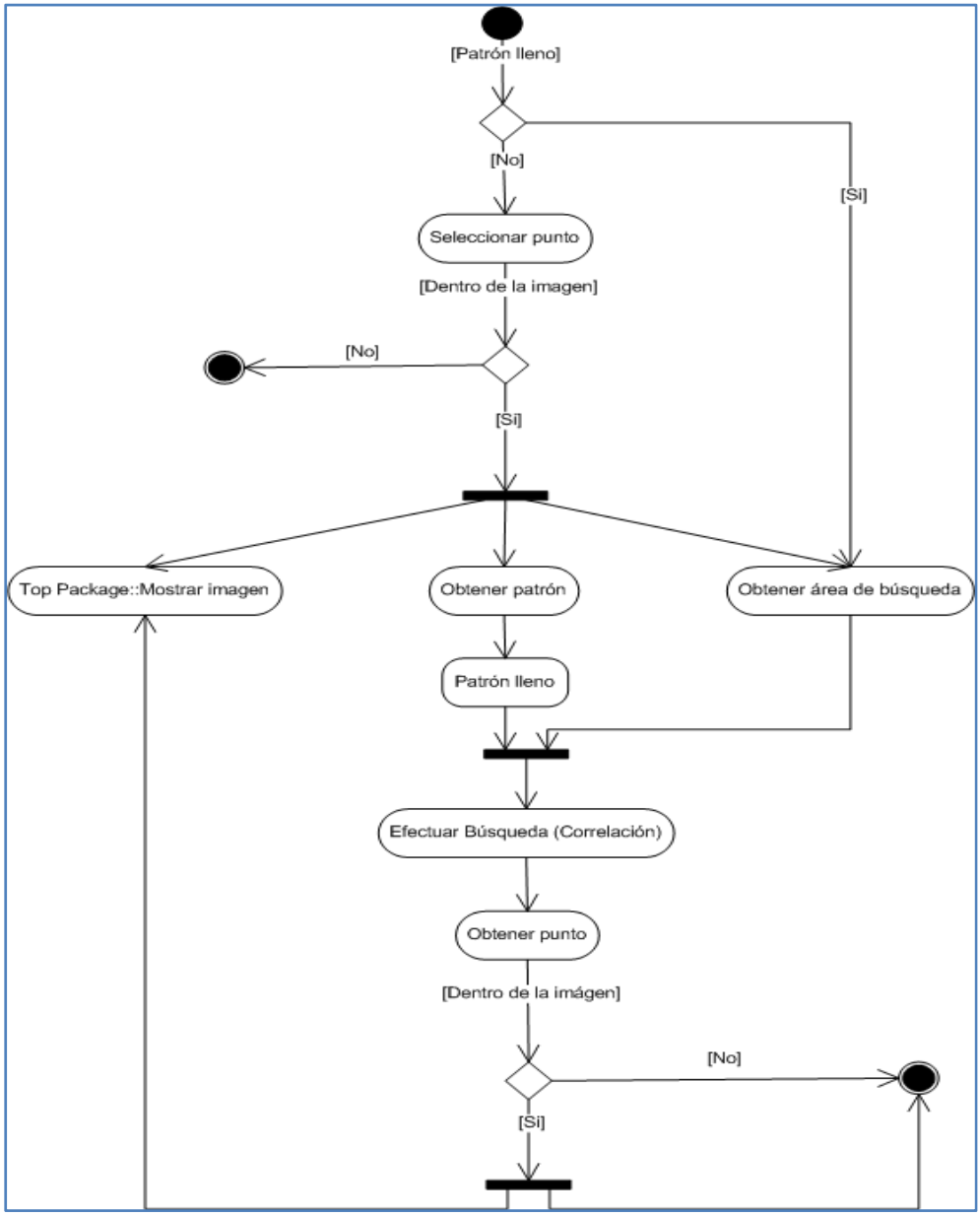


Figura 3.5 Diagrama de actividades correspondiente al caso de uso “Seleccionar objeto”

Para que el sistema entre en el estado de selección de objeto (figura 3.5) es necesario que el usuario de un "clic" con el botón izquierdo del mouse para seleccionar un punto dentro de los límites de la imagen. Una vez que se validó la ubicación del "clic" del usuario se procede a realizar tres actividades, la primera es mostrar la imagen ya con el objetivo seleccionado; la segunda se refiere a llenar la matriz patrón con los pixeles del objeto de interés; la tercera corresponde a llenar la matriz de búsqueda del área de vecindad en donde se calcula que esté el objetivo. Después de completadas estas últimas dos actividades se efectúa la búsqueda mediante el algoritmo de correlación, el cual arroja como resultado las nuevas coordenadas del objeto de interés, las cuales son validadas para después mostrar la imagen correspondiente con el objeto en su nueva ubicación.

Repitiendo el proceso anterior se logra hacer el seguimiento del objeto. Nótese que existe un caso de uso llamado "Deseleccionar objeto" el cual se activa con un "clic" con el botón derecho del mouse, para el cual no se hace un diagrama de actividades ni se incluye su descripción en ningún otro. Esto se debe a que la función del caso de uso es muy simple que se consideró innecesario tratarlo aparte. Basta con saber que una vez que un objeto es seleccionado y se le está haciendo seguimiento, el usuario puede o no cancelar la selección del objeto, acción de la que depende que se continúe realizando seguimiento o sólo se muestren capturas de la cámara.

A continuación se muestra el diagrama de actividades correspondiente al caso de uso “Cambiar nivel de exposición”.

Figura 3.6 Diagrama de actividades para el caso de uso “Cambiar el nivel de exposición”

Las actividades que se llevan a cabo para aumentar o reducir el nivel de exposición a la luz son muy simples, mientras el sistema se encuentra en el proceso de captura de imágenes y/o seguimiento de un objeto, el usuario puede cambiar el valor correspondiente al nivel de exposición mediante el uso de una barra de desplazamiento, con lo cual los cambios se hacen evidentes inmediatamente y el usuario puede continuar con el uso del sistema.

Actividades y explicación análogas son las que se requieren para cambiar el tamaño de la región patrón que contiene al objeto de interés. Su diagrama de actividades es el que se muestra a continuación.

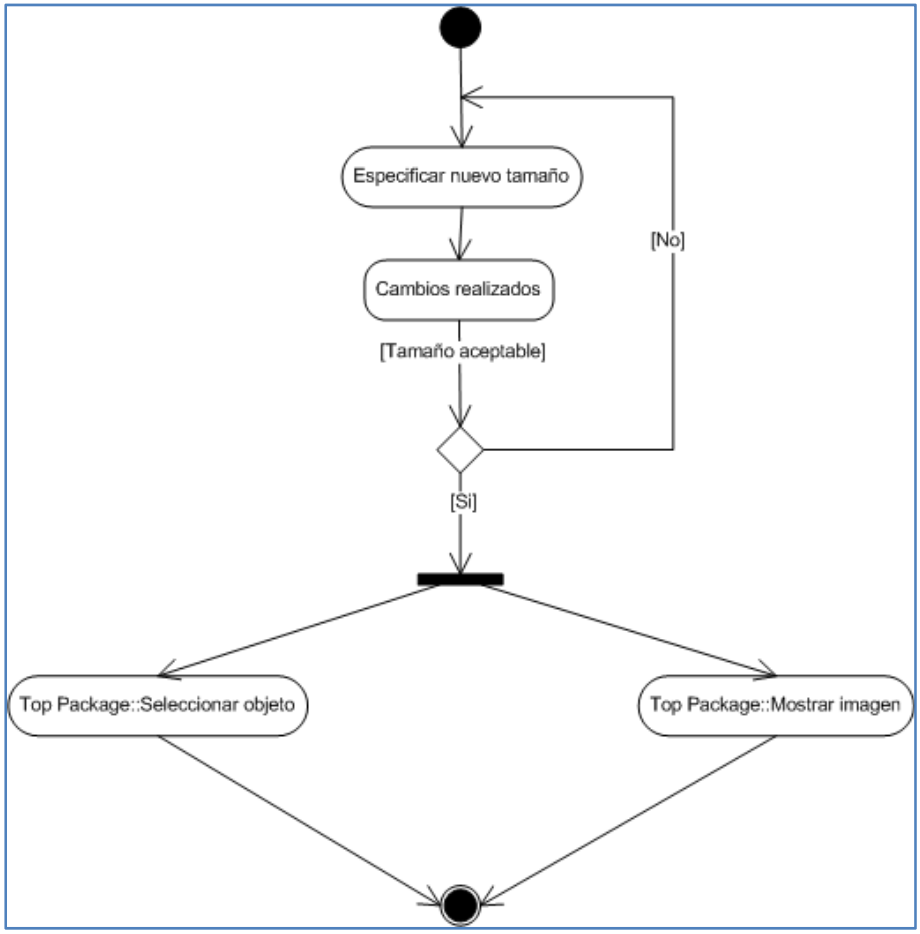


Figura 3.7 Diagrama de actividades para el caso de uso “Cambiar tamaño región patrón”

Por último, para el caso de uso en el que el usuario tiene la posibilidad de guardar una secuencia de imágenes se obtienen las siguientes actividades.

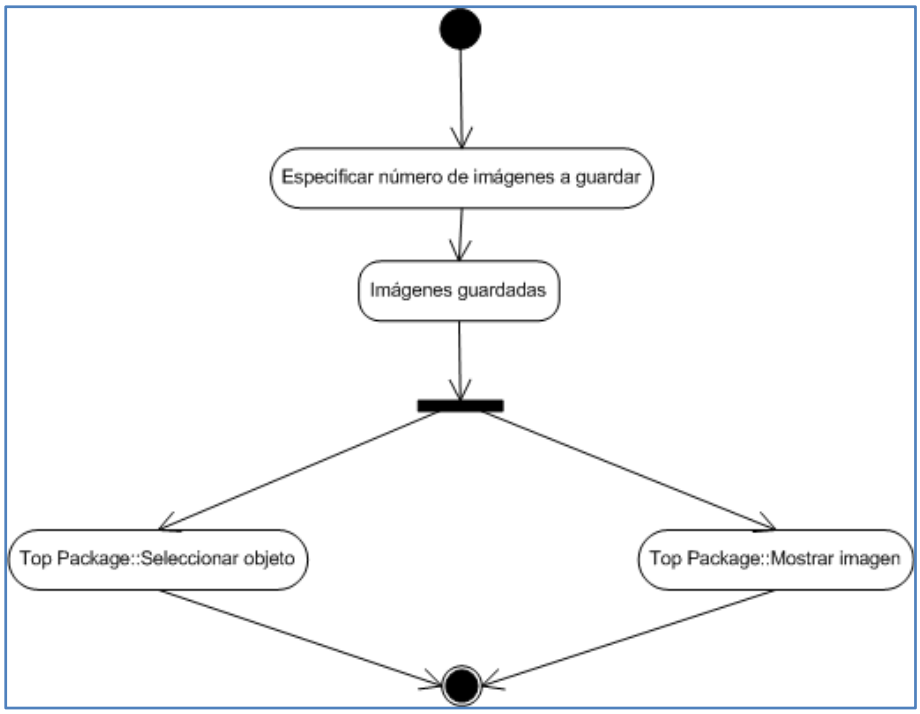


Figura 3.8 Diagrama de actividades para el caso de uso “Guardar secuencia”

De la imagen anterior se observa que el usuario puede guardar una secuencia de capturas en el momento que él lo deseé, solo basta con indicar en un campo de texto el número de imágenes a guardar y presionar un botón para terminar la acción.

Capítulo 4

Diseño del sistema

4.1 Diseño

Una vez que se ha dejado claro el funcionamiento del sistema se procede al diseño de sus componentes integrales, desde el punto de vista del desarrollo del software, por lo que es necesario el diseño de los objetos, sus atributos y acciones que los formarán.

Para esta tarea se empleará una herramienta UML que son los "Diagramas de secuencias", estos diagramas ayudan a describir la creación e interacción de los objetos a lo largo del tiempo por lo que son de mucha ayuda para una buena definición de las interacciones y posibles excepciones que surjan de la operación del sistema.

A continuación se muestra una serie de diagramas de secuencias, los cuales corresponden a los escenarios más importantes de los procesos clave del sistema. En términos del objetivo de este trabajo, la inclusión de los diagramas de excepciones y escenarios alternativos no resulta crítica.

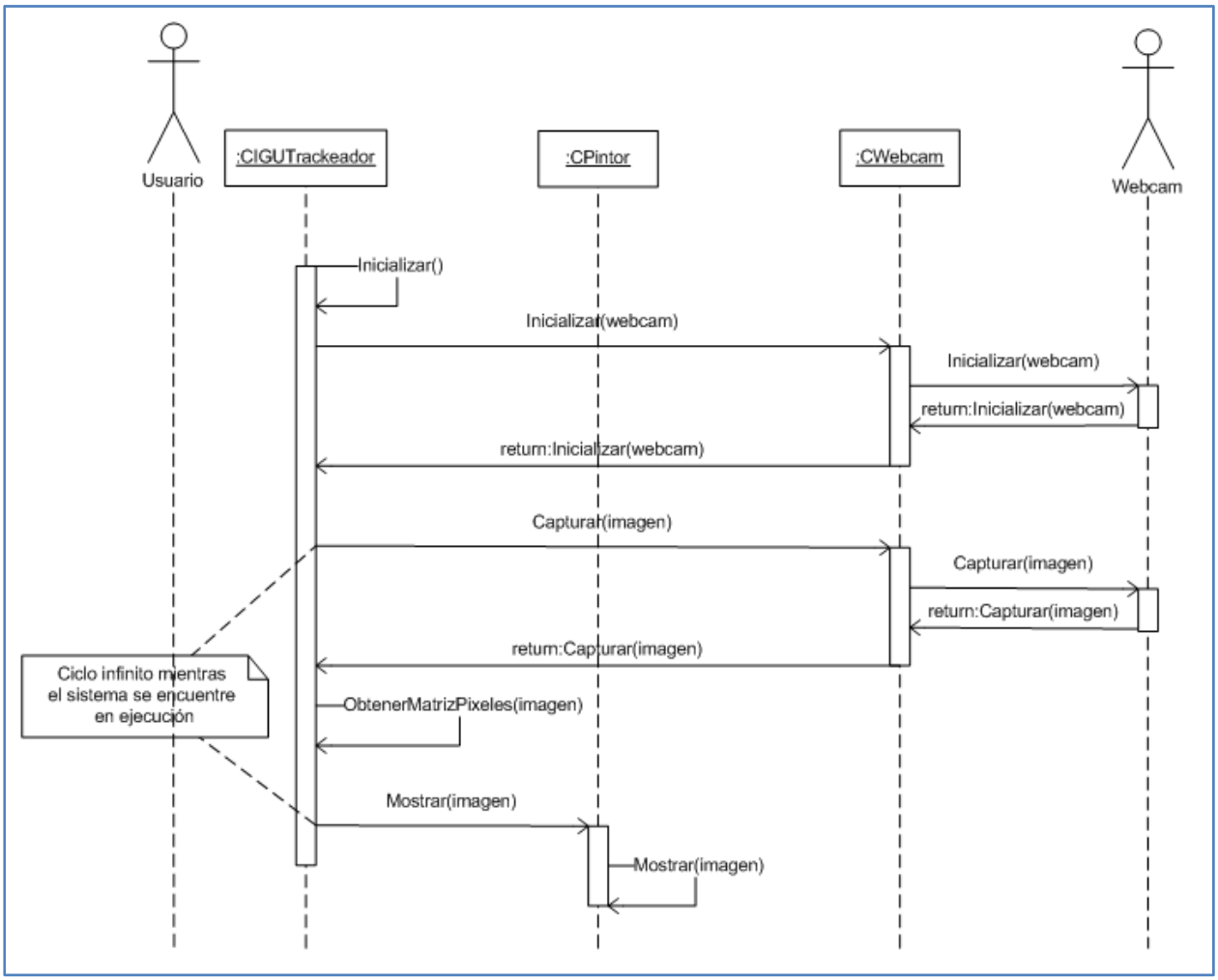


Figura 4.1 Diagrama de secuencia de la captura cíclica de imágenes

La figura 4.1 muestra la secuencia necesaria para la captura y despliegue de imágenes, la cual se inicia apenas entra en ejecución el sistema y finaliza con el cierre del mismo.

La figura 4.2, que a continuación se muestra, corresponde al escenario de selección de objeto para darle seguimiento. Para entrar en este escenario el usuario tiene que dar "clic" en un objeto de la secuencia de imágenes que se le muestra, con lo que el sistema guardará la coordenada del "clic" del mouse relativa a la imagen mostrada para luego realizar el algoritmo de correlación usando ventanas de búsqueda y patrón formadas a partir de las coordenadas del objeto de interés.

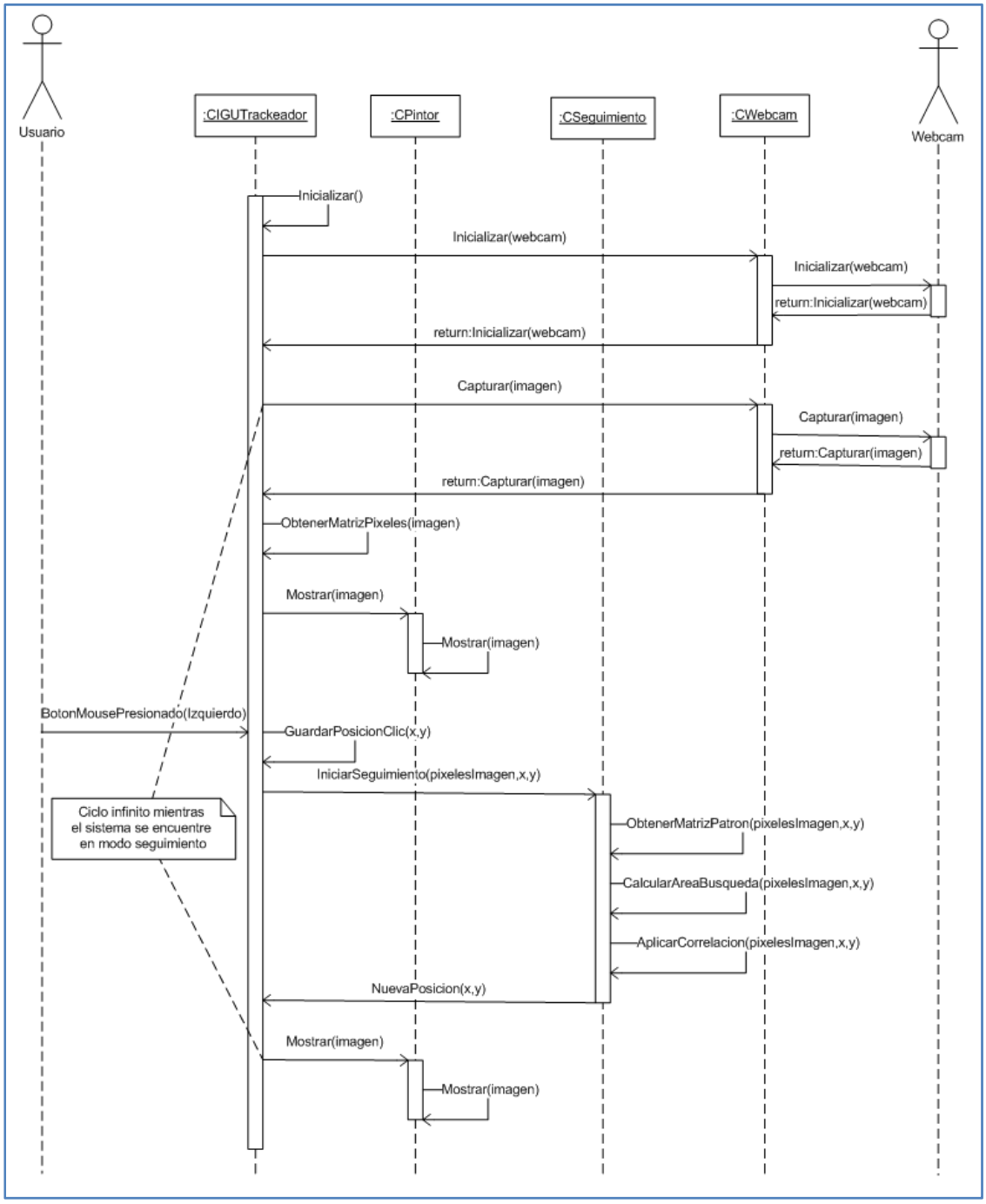


Figura 4.2 Diagrama de secuencia del escenario de selección de objetivo y seguimiento de objetos

La secuencia de eventos que son necesarios para cambiar el nivel de exposición de la cámara a la luz se inicia cuando el usuario usa una barra de desplazamiento que controla el nivel de luz que recibe la webcam, con lo el sistema almacena el nuevo valor y envía una petición a la webcam para cambiar el nivel actual por el nuevo valor y posteriormente se muestra la imagen ya con el nuevo nivel de exposición lumínica.

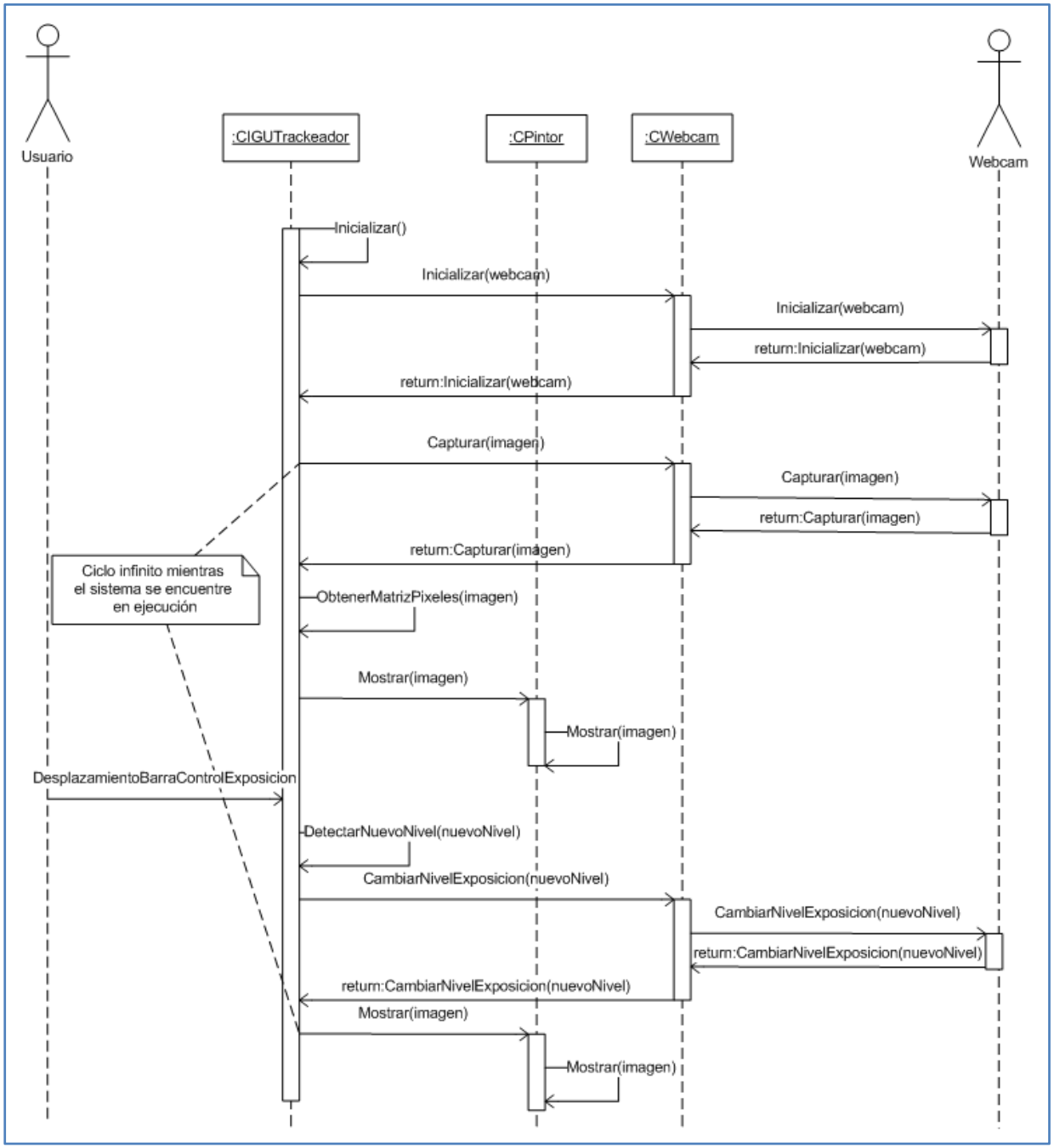


Figura 4.3 Diagrama de secuencia del escenario de cambio de nivel de exposición a la luz

Para cambiar el tamaño de la región patrón el sistema deberá estar en modo de sólo captura/muestra de imágenes. Se inicia cuando el usuario usa la barra de control que determina el tamaño de la zona patrón, a lo que el sistema reacciona actualizando el dicho tamaño. Nótese que los cambios no serán visibles hasta que el usuario de clic en una zona de la imagen, lo cual pondría al sistema en modo seguimiento. En la figura 4.4 se ilustra lo mencionado.

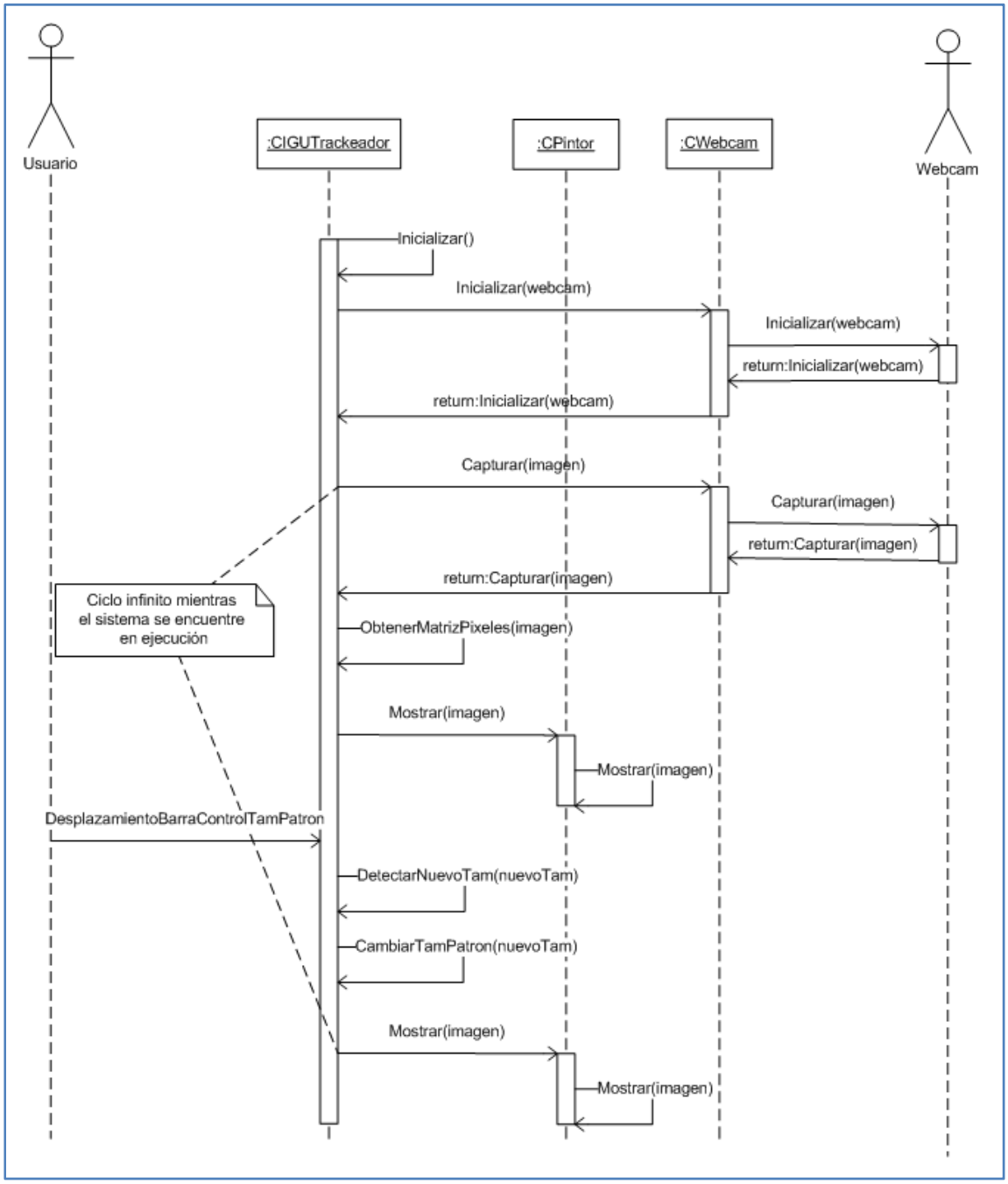


Figura 4.4 Diagrama de secuencia del escenario de cambio de tamaño de la región patrón

Finalmente, si el usuario desea almacenar una secuencia de imágenes tendrá que introducir en el cuadro de texto correspondiente el número que corresponda a la cantidad deseada de imágenes que se guardarán, acto seguido deberá dar clic en un botón. A partir de ese momento el sistema empezará a guardar y capturar imágenes cíclicamente hasta que la cantidad de imágenes requeridas se haya completado. Esto proceso se aprecia en la figura 4.5

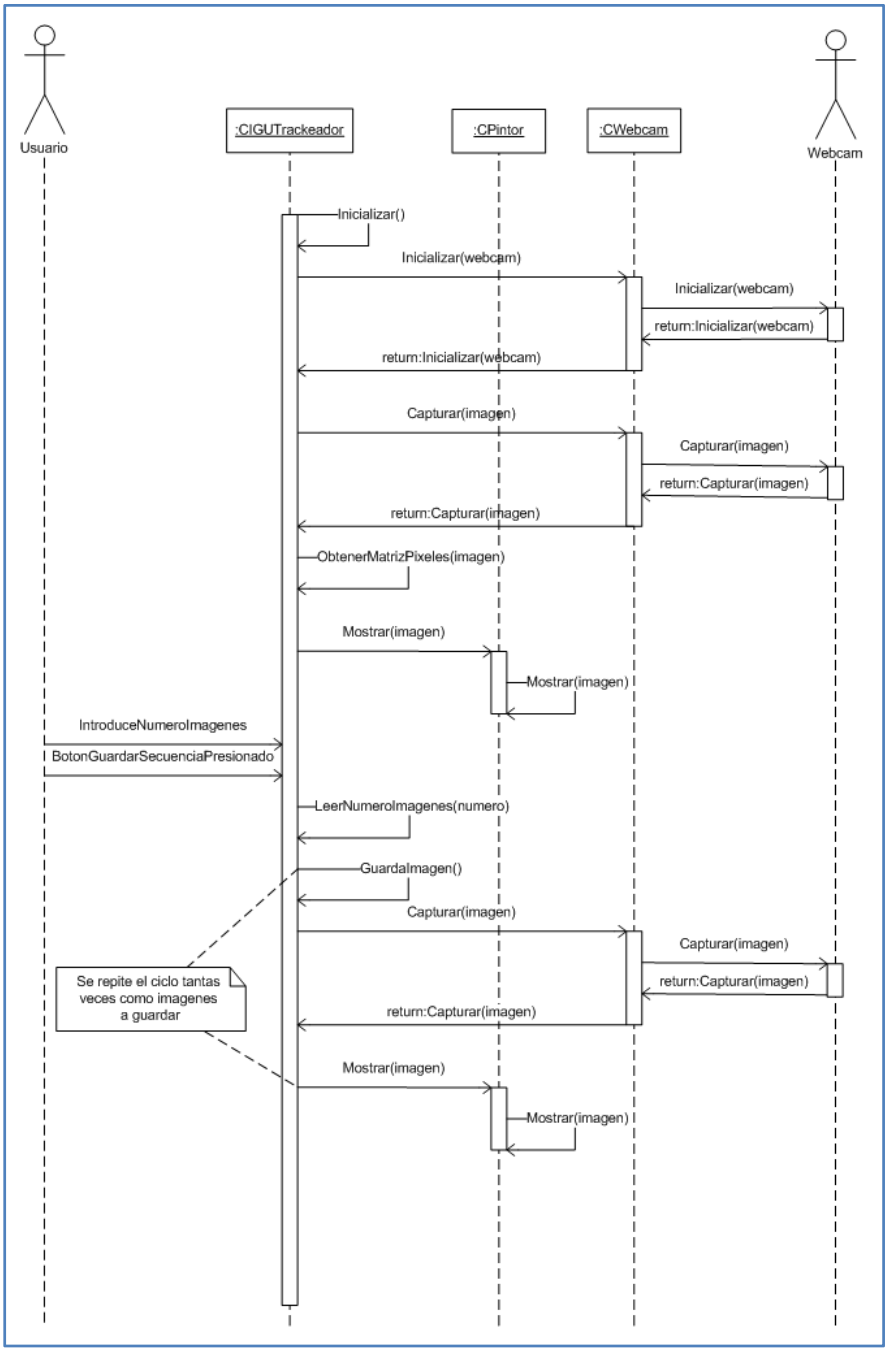


Figura 4.5 Diagrama de secuencia que muestra el escenario para guardar sucesión de imágenes

Para describir de forma más precisa el conjunto de funciones generadas de acuerdo a las necesidades de cada uno de los escenarios que constituyen el funcionamiento del sistema, así como sus propiedades se hará uso de un "Diagrama de clases", el cual será de mucha ayuda para generar la estructura del código final de la aplicación.

En el diagrama de clases de la figura 4.6 se resumen las principales clases del sistema, la forma en que se relacionan y los atributos y funciones de cada una.

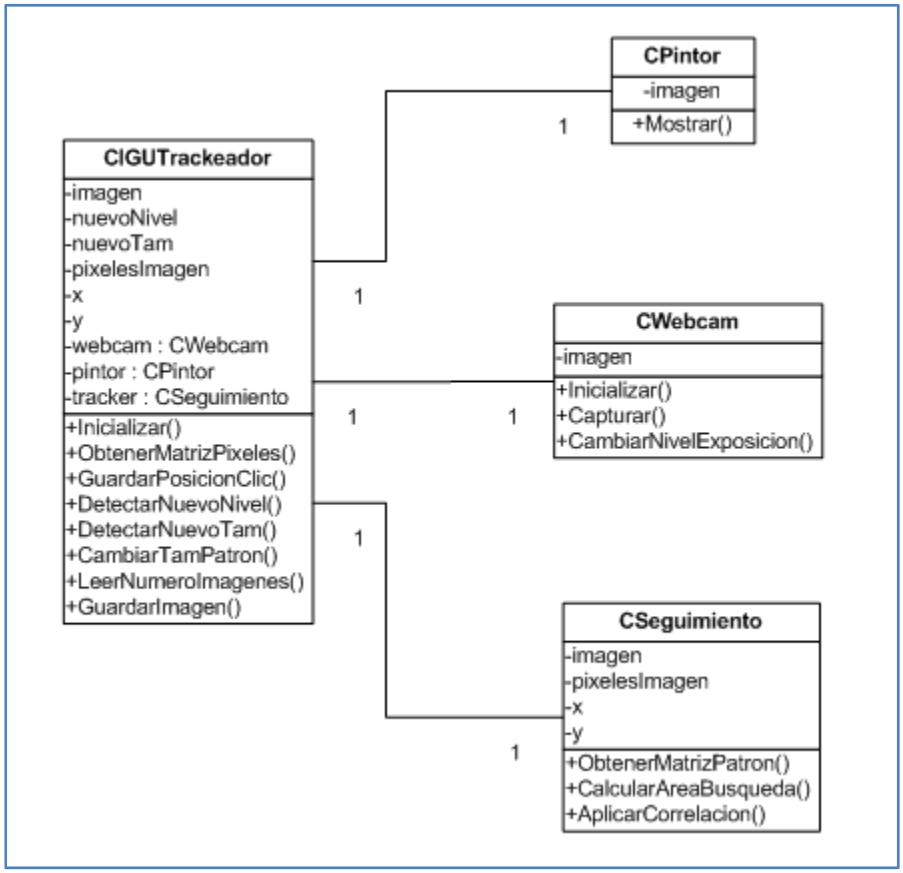


Figura 4.6 Diagrama de clases del sistema

4.2 Video4Linux2

Como ya se ha mencionado, se usará como dispositivo de captura una webcam USB, lo cual no es nada nuevo, pues se pueden encontrar varios proyectos en los que se trabaja usando este tipo de dispositivos, pero la mayoría de ellos tienen la peculiaridad de estar montados en sistemas Windows, sistema operativo para el cual están orientados la mayoría de los controladores proporcionados por los fabricantes, además de que este sistema operativo cuenta con diversas API's (del inglés *Application Programming Interface*) como *DirectShow*, *QuickTime*, o *VideoForWindows*, que son un conjunto de funciones residentes en bibliotecas (generalmente dinámicas, también llamadas *DLL's* por sus siglas en inglés) que permiten a los programadores crear código que utilice dispositivos multimedia (entre otros) que normalmente no son creados con este fin. Por ejemplo, una webcam no está diseñada ni es comercializada para ser usada como sustituto de una cámara de vigilancia. Es por eso que los fabricantes no ofrecen al consumidor este tipo de herramientas (API) pues consideran que sus productos se usarán única y exclusivamente con los controladores del fabricante y las limitantes de uso que estos les impongan. De cualquier forma y dado que por lo general los fabricantes en ningún lado especifican que el uso de sus productos solo esta licenciado con los drivers que ellos proveen, en este trabajo se hace uso de una webcam como medio de adquisición de imágenes, por lo tanto es necesario un mecanismo para acceder a las prestaciones de dicho dispositivo, es decir, se requiere una API.

Una de las características requeridas por el sistema que en este trabajo se propone es el uso de GNU/Linux como sistema operativo, para el cual existe una API muy poderosa llamada *Video4Linux* (abreviada como *V4L*) la cual se describe de manera breve a continuación.

V4L está desarrollada en lenguaje C, proporciona el conjunto de instrucciones necesarias para la captura de video en entornos GNU/Linux. Muchas webcams USB, sintonizadoras de TV y otros periféricos son soportados. *Video4Linux* fue integrado al núcleo Linux desde la versión 2.1.x del kernel.

Actualmente *V4L* se encuentra en su segunda versión, denominada *Video4Linux2* (*V4L2*), la cual arregla algunos fallos de su antecesora y ha sido incluida desde la versión 2.5.x del núcleo Linux, sustituyendo a la primer versión y dejando un modo de compatibilidad para las aplicaciones *V4L*;

aun así el soporte puede ser incompleto y los creadores de la API recomiendan usar hardware *V4L2* en modo *V4L2*.

En la figura 4.7 se muestra un esquema general del funcionamiento de *V4L2* para dispositivos de captura.

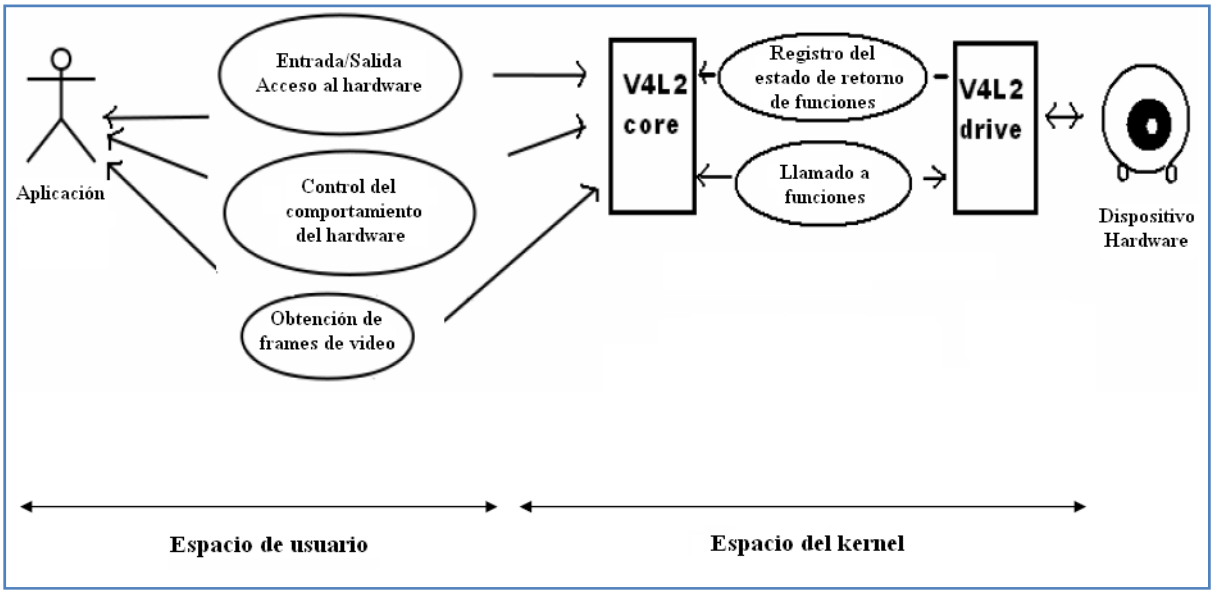


Figura 4.7 Esquema de funcionamiento de V4L2 en dispositivos capturadores

En términos simples, al usar el API *V4L2* para establecer comunicación con un dispositivo se está creando un driver o controlador para la aplicación que se está programando. En GNU/Linux todo es un archivo, así que los usuarios pueden controlar dispositivos a través de operaciones como *open(), read(), write()* e *ioctl()*, que es lo mismo que hacen los drivers provistos por los fabricantes.

La llamada al sistema *ioctl()* significa literalmente "i/o control" y sirve para controlar dispositivos, pero diferentes dispositivos tienen diferentes funciones que pueden ser usadas mediante *ioctl*, por ejemplo, en una impresora se podrían controlar funciones como la elección de la tinta a usar (b/n o a color), la calidad de la impresión, impresión a doble cara, cancelar impresión, etc. mientras que en un dispositivo CD/WR se podrían controlar la velocidad de grabación, el inicio y finalización de la grabación o la apertura de la bandeja de disco. Esto es

gracias a que llamadas al sistema como *ioctl* cuentan con un argumento en el que se especifican las funciones del dispositivo de las que se hará uso.

Una característica de los dispositivos de video es que tienen muchas configuraciones diferentes, como el tamaño de los frames, el *frame rate* (velocidad con que se muestran las imágenes), el formato de la imagen, brillo, etc. Si cada controlador de dispositivo tuviera su propia versión de *ioctl* sería difícil para las aplicaciones soportar gran variedad de dispositivos. Esta es la causa por la que *V4L* existe, para estandarizar las peticiones que se pueden hacer y las estructuras para transferir información que estas necesiten.

V4L es un subsistema de kernel que sirve de puente o vínculo entre el espacio de usuario y el espacio del kernel. En él se reciben las llamadas *ioctl* hechas por el usuario, se procesan y se mandan al dispositivo para su ejecución.

Los pasos para programar un dispositivo *V4L2* son:

- Abrir el dispositivo
- Cambiar las propiedades del dispositivo (por ejemplo, seleccionar un estándar de video, definir el nivel de brillo, etc.)
- Negociar un formato para los datos
- Negociar un método de entrada/salida
- Repetir el uso del método E/S adecuado
- Cerrar el dispositivo

Información más detallada del uso de la API *Video4Linux2* puede ser encontrada en [10].

4.3 LiveCD/DVD

Las razón por la cual se decidió montar el sistema resultante de este trabajo en un LiveDVD es que se dotaría al sistema con un cierto grado de portabilidad ya que no sería necesario tener una computadora especializada para labores de seguimiento, pues sólo bastaría con introducir el DVD en una computadora que cumpla con algunas especificaciones (ver requisitos en el cap. 1) y conectar una webcam para poder utilizar el sistema de seguimiento.

Pero… ¿Qué es un LiveCD/DVD, como se usa y como se crea? Las respuestas a esas preguntas se exponen a continuación.

Un liveCD o liveDVD es un sistema operativo (generalmente acompañado de aplicaciones) almacenado en un medio extraíble, tradicionalmente un CD o un DVD (de ahí sus nombres), que puede ejecutarse desde éste, sin necesidad de ser instalado en el disco duro de una computadora, para lo cual usa la memoria RAM como disco duro virtual y el propio medio como sistema de ficheros. De hecho no se requiere siquiera que la computadora tenga un disco duro instalado.

La palabra “live” se deriva de que estas distribuciones son un espejo completo y ejecutable (“en vivo”) de un sistema operativo.

Para usar un liveCD/DVD es necesario configurar el BIOS (Sistema Básico de Entrada y Salida) de la computadora para que arranque desde la unidad lectora de CD/DVD, reiniciando luego la computadora con el disco dentro de la lectora para que el liveCD se inicie automáticamente.

En el arranque, se le pueden dar distintos parámetros para adaptar el sistema a la computadora, como la resolución de pantalla o bien activar o desactivar la búsqueda automática de determinado hardware.

La mayoría de estas distribuciones usan un sistema operativo basado en el núcleo Linux, pero también se usan otro sistemas como BeOS, FreeBSD, Minix, Solaris, OS/2 o incluso Microsoft Windows (sin embargo, distribuir un liveCD de este último se considera ilegal).

El primer Live CD Linux fue Yggdrasil Linux en 1995, aunque fue poco exitosa. Posteriormente surgió DemoLinux (año 2000).

El auge de esta modalidad de Linux se inició alrededor del año 2003 con la distribución alemana de Knoppix, basada, a su vez, en la distribución de software Debian. Una de las mejoras de este método fue la compresión cloop, esto permitió sobrepasar los 650-700 MB del CD (se usaba el driver loop) y lograr introducir hasta 2 GB. La asociación española de Hispalinux ha popularizado un sistema propio, denominado Metadistros fundamental en las distribuciones Linex y Guadalinex.

Uno de los mayores inconvenientes que hubo de este enfoque es el requerimiento de una gran cantidad de memoria RAM (aunque 256 MB son más que suficientes y hay distribuciones que funcionan perfectamente en 128), una parte para su uso habitual y otra para funcionar como el disco virtual del sistema, inconveniente que ha desaparecido casi por completo con la disminución en los precios y aumento de la capacidad de la menoría RAM. Otra característica inconveniente es que por lo general los cambios que se hagan a la computadora (sistema operativo o archivos de usuario) son volátiles y desaparecerán apenas se apague la computadora o se le quite la corriente, es decir, la computadora vuelve al estado anterior a la inserción y carga del liveCD/DVD, aunque hay versiones que pueden almacenar preferencias si así se desea.

Las ventajas que presentan este tipo de medios es que, como ya se había mencionado, no hay instalación (a menos que se desee, pues hay algunos liveCD/DVD incluyen una herramienta que permite instalarlos en el disco rígido), por lo que no hay que tocar el disco duro ni seguir procedimientos complicados (habituales en muchas distribuciones Linux). Además, los datos, particiones o sistemas operativos existentes en el disco duro no se pierden. Suelen tener un reconocimiento de hardware avanzado, fruto también de las últimas versiones del kernel Linux (2.6.x).

Existen varias formas para crear un liveCD/DVD a partir de un sistema operativo funcional, muchas de las cuales contemplan procesos un tanto difíciles y riesgosos para el sistema en que se basen, pues éste podría quedar inaccesible. Debido a lo anterior es que en este trabajo se optó por usar una herramienta que, una vez instalada, facilita la creación versiones liveCD/DVD de sistemas Klikit-Linux, Ubuntu (KUbuntu, EdUbuntu, XUbuntu) y sus derivados (como Linux Mint) a partir de una instalación del sistema en el disco duro de la computadora. El nombre de esta herramienta es *Remastersys*.

Remastersys permite crear 2 tipos de imágenes (*.iso*):

- Una imagen completa del disco duro incluyendo los datos de la carpeta */home* de los usuarios existentes.
- Una imagen completa del disco duro pero sin incluir los datos de la carpeta /home. Esta opción se usa para redistribuir el sistema con otras personas sin compartir información privada.

Para lo cual bastaría teclear lo siguiente en una consola con privilegios de súper-usuario (root):

```
# remastersys backup
```

Si queremos hacer una imagen completa de la instalación que hay en el disco duro.

O bien:

```
# remastersys dist
```

Para hacer una imagen sin datos privados de los usuarios

El tiempo que tomará el proceso de creación de la imagen estará en función de la velocidad del procesador y la cantidad de memoria RAM con la que cuente el equipo.

Una vez finalizada la creación de la imagen del sistema operativo sólo hará falta grabarla con cualquier aplicación de grabación de CD/DVD como Nero, K3B, Brasero o cualquiera que soporte grabación de imágenes *.iso*.

El medio a utilizar dependerá del tamaño de salida del archivo de imagen, si es menor a 700 MB se usará un CD, pero si es mayor se tendrá que usar un DVD.

Por último se debe mencionar que el tamaño de la imagen resultante dependerá del tamaño de la instalación en la cual esté basada.

Mayor información sobre el uso de *Remastersys* puede ser encontrada en [11].

Capítulo 5

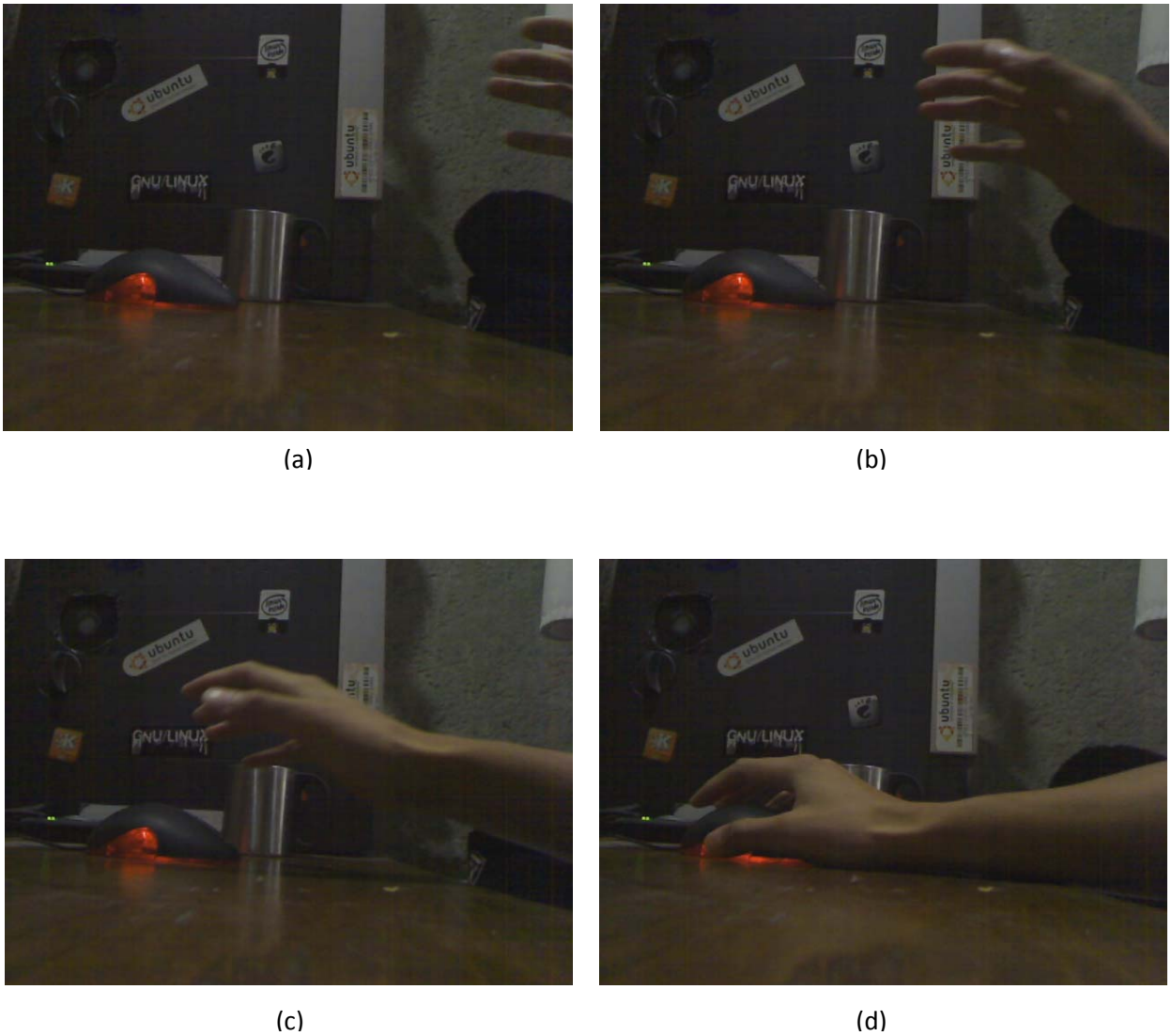
Resultados

5.1 Captura de imágenes

Como primer paso para realizar cualquier tipo de trabajo con imágenes digitales en un sistema de visión artificial se debe tener un mecanismo de captura de imágenes suficientemente eficaz en términos de velocidad de captura y del tiempo en que tardan las imágenes en ser mostradas, pues obtener malos resultados en esta etapa afectaría directamente el desempeño del sistema en general, aún cuando se usaran algoritmos muy avanzados para el seguimiento.

En este proyecto se usó una cámara con velocidad de captura de imágenes de 30 fps en resolución 640x480 pixeles, lo cual es aceptable considerando que no se trata de una cámara especializada.

A continuación se muestran imágenes que forman parte de una secuencia con el fin de ilustrar que se cumplió con el objetivo de calidad y velocidad en la captura de imágenes.



(a) (b)

(c) (d)

Figura 5.1 Muestra de una secuencia de imágenes capturadas por el sistema

Obsérvese que no son imágenes contiguas dentro de la secuencia, en vez de eso se optó por mostrar imágenes en las que hicieran evidente la ocurrencia de movimiento en escena. Lo mismo sucede con las secuencias siguientes.

5.2 Selección y seguimiento de objetos

El seguimiento de objetos móviles es el principal problema a resolver en esta tesis y para lo cual se desarrolló un sistema de seguimiento basado en un algoritmo de correlación.

Se hicieron pruebas de desempeño de la aplicación en distintos entornos de iluminación sin hacer ningún cambio al nivel de exposición a la luz de la lente de la cámara, así mismo se hicieron pruebas de seguimiento a objetos con distinta velocidad de movimiento, en los cuales se obtuvieron resultados que llevan a enunciar lo siguiente:

- Las condiciones de iluminación afectan el desempeño del sistema, pues al haber poca luz que ilumine el entorno en el que se realice el seguimiento el objeto de interés tiende a confundirse con su ambiente adquiriendo tonalidades oscuras que hacen imposible encontrar diferencias en las características de los pixeles. Lo cual podría cambiar si se usan filtros para mejorar las imágenes antes de aplicar el algoritmo de seguimiento sobre ellas, aspecto que no es tratado en esta tesis (ver trabajo futuro).
- Si las condiciones de iluminación son adecuadas se puede realizar seguimiento sobre objetos en los que su movimiento no sea caótico, o al menos no presentar velocidades y trayectoria aleatorias y/o desiguales de gran magnitud. Los movimientos deben ser de dirección y baja velocidad. Otro aspecto que afecta el desempeño del sistema es la aparición de oclusiones en las que el objeto desaparezca totalmente o por un periodo prolongado de tiempo, ya que el sistema perderá la ubicación del objetivo si este reaparece en una posición más allá de la ventana de búsqueda.

De manera general se puede decir que el objetivo de realizar un sistema de seguimiento de objetos fue alcanzado satisfactoriamente aunque puede mejorar con el trabajo futuro.

A continuación se muestra una secuencia de imágenes que ilustra el seguimiento de objetos. En este caso el patrón a seguir (recuadro interior rojo) es una zona oscura dentro de la pantalla de un teléfono celular, la cual se busca dentro del recuadro exterior verde (ventana de búsqueda).

Se observa que si el objeto presenta movimientos bruscos el sistema no será capaz de mantener su ubicación.

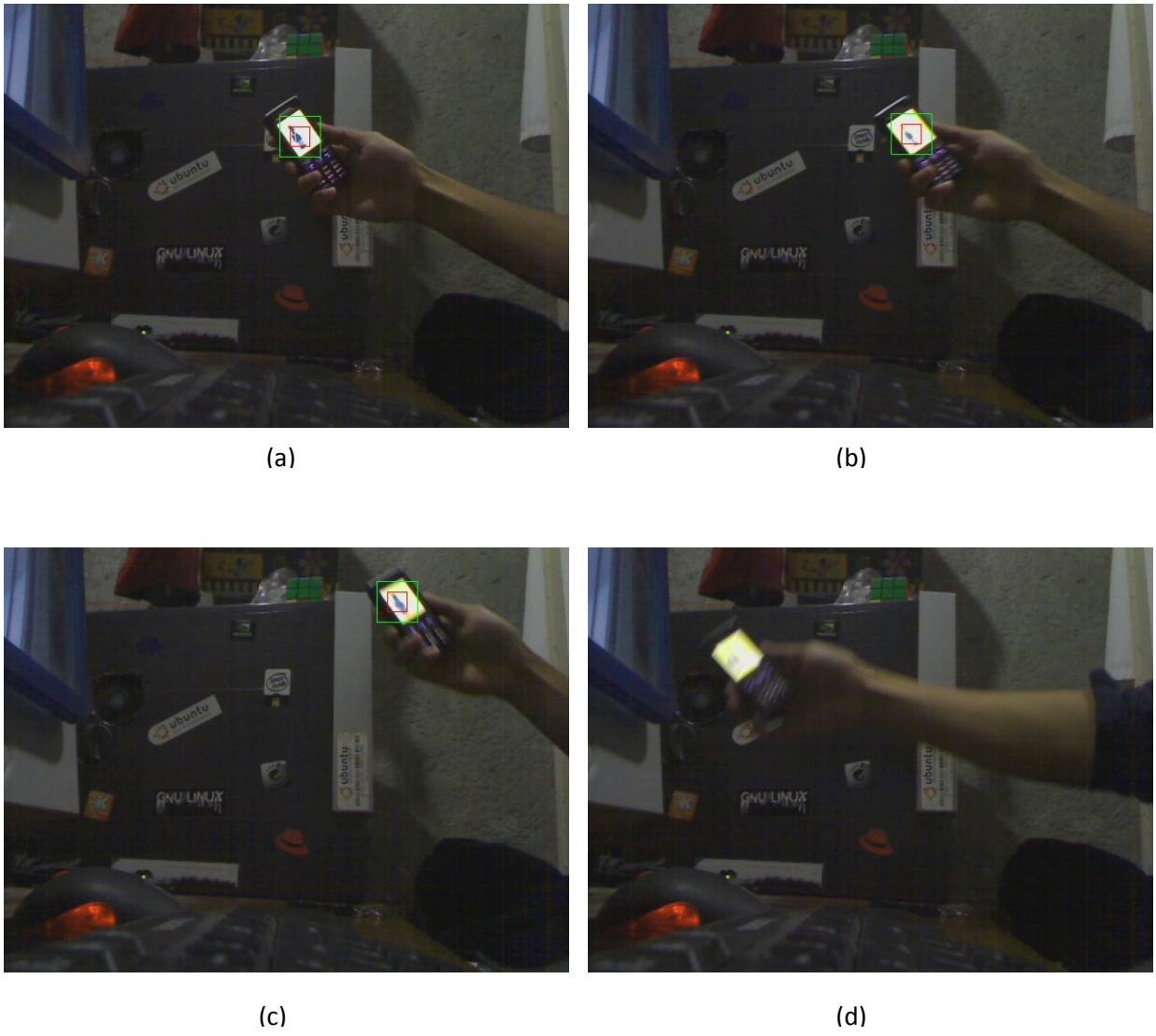


Figura 5.2 Seguimiento de un objeto que presenta movimientos bruscos

5.3 Cambiar nivel de exposición

Para combatir el problema de escenarios en los que las condiciones de iluminación no puedan ser controladas para obtener un buen funcionamiento del algoritmo de seguimiento, se ha dotado al sistema con un control que regula el nivel de luz captado por la cámara (exposición). Sin embargo el exceso en el nivel de luz captado por la lente afecta directamente la velocidad de captura de la cámara, así que es recomendable hacer sólo los ajustes mínimos necesarios para mejorar las imágenes capturadas.

A continuación se muestra una secuencia en la que el entorno es muy oscuro por lo que se hace uso del control de nivel de exposición para mejorar las condiciones para el seguimiento.

Figura 5.3 Utilización del nivel de exposición para mejorar las condiciones de iluminación

5.4 Cambiar el tamaño de la ventana patrón

En ocasiones, cuando el objeto de interés sea más grande que el tamaño de la ventana de búsqueda o exista mucha similitud entre los colores del entorno y del objetivo, será necesario ajustar dicho tamaño a manera de que el objeto pueda ser enmarcado dentro de la zona de búsqueda. Para ese propósito se cuenta con un control que regula el radio de la ventana patrón, el cual también cambia el tamaño de la ventana de búsqueda de manera proporcional.

Este control ha sido implementado en la interfaz de usuario y para activarlo sólo es necesario usar una barra de control mientras se está en modo de captura/muestra de imágenes. Para visualizar que los cambios surtan efecto será necesario dar clic en un objetivo para que los recuadros de selección de objetivo se vuelvan a mostrar.

En modo seguimiento, entre mayor sea el tamaño de la ventana patrón mayor será el número de regiones candidatas que cabrán dentro de la región de búsqueda y por ende mayor será el tiempo que tardará el algoritmo de correlación en determinar el mejor candidato. Esto afecta directamente el repintado de la imagen ya que esta no puede ser actualizada hasta que no haya otra disponible, y las imágenes no se consideran disponibles hasta no haber pasado por el algoritmo de correlación.

5.5 Guardar secuencias de imágenes

La opción de guardar un número determinado de imágenes del escenario en cuestión es útil cuando se realizarán detenidamente otro tipo de estudios sobre el objeto de interés. Las secuencias mostradas anteriormente en este capítulo fueron capturadas usando la opción correspondiente con que cuenta el sistema. Es por eso que se considera cubierto este objetivo e innecesaria una nueva captura para este apartado. Sólo es necesario añadir que las capturas realizadas se guardan dentro de una carpeta llamada "*Secuencias*" que se encuentra en el escritorio del sistema LiveDVD

5.6 Sistema en LiveDVD

La inclusión y utilización del sistema de seguimiento de objetos en un LiveDVD fue llevada a cabo (ver sección 4.1) y probada con éxito en dos equipos de cómputo distintos:

- Una laptop HP Pavilion zd700 con:
  - Procesador Intel Pentium IV a 1.5 GHz
  - 1 GB de memoria RAM
- Una computadora ensamblada con:
  - Procesador Intel Pentium IV a 2.6 GHz
  - 1 GB de memoria RAM

Adicionalmente, el sistema de seguimiento en LiveDVD fue probado en la computadora en la que se desarrolló, la cual cuenta con las siguientes características:

  - Procesador Intel Pentium IV a 3.0 GHz
  - 1 GB de memoria RAM

Por lo que se considera que el sistema funcionará adecuadamente en computadoras que tengan recomendablemente 1 GB de memoria RAM y procesador Intel superior a 1.5 GHz o similar.

Conclusiones

Los sistemas de visión tienen mucha importancia y cada vez son aplicados en muchas actividades cotidianas que no requieren gran cantidad de recursos, por lo que es necesario buscar sistemas alternativos que utilicen los mismo métodos usados en los grandes sistemas de seguimiento pero dándoles un enfoque, hasta cierto punto, "casero", pues no se requiere mayor inversión que la necesaria para un sistema de cómputo (con el cual es muy común contar) y un dispositivo de captura. Además el uso de un medio no destructivo de información, como lo es un LiveDVD, aportan una característica deseada por personas que no desean alterar el uso de su computadora personal. Con lo que se consideran alcanzados los objetivos mencionados al inicio de este documento, sin dejar de lado la capacidad de crecimiento del sistema, claro está, pues con la incorporación de algoritmos adicionales que cubran las limitaciones de este proyecto, se obtendría un sistema de seguimiento portátil mucho más eficaz y aplicable a situaciones bastante más complejas.

En resumen, se pueden considerar los siguientes puntos como las conclusiones más relevantes:

- La elaboración de este proyecto involucró la aplicación de conocimientos adquiridos a lo largo de una carrera de estudios profesionales, tales como el adecuado análisis diseño e implementación de sistemas de cómputo; utilización (en la codificación) de los mecanismos de programación aprendidos durante los estudios, así como el uso y aplicación de las imágenes digitales.
- Durante la investigación, estudio y desarrollo de la aplicación, se determinaron los componentes y características que constituyen al sistema y sus capacidades, con lo que se adquirió experiencia acerca de los principales problemas a resolver y los aspectos de los cuales depende el desempeño y alcance de los sistemas de visión artificial.

- El sistema obtenido es pequeño pero completamente funcional, además significa un ahorro de recursos al no requerir de la utilización de hardware especializado.

- Finalmente y de mayor importancia, se creó una aplicación que tiene capacidad de expansión y será mejorada para afrontar nuevos retos, con lo que se comprendió que en el ámbito computacional nada está dicho y que se debe permanecer en constante adquisición y actualización de conocimientos y habilidades para poder ser aplicados a la resolución de problemas reales, cotidianos y específicos.

Trabajo futuro

El trabajo pendiente de realizar esta orientado a solucionar o proveer capacidades al sistema de seguimiento realizado. A continuación se mencionan las carencias que fueron detectadas a lo largo del desarrollo de este trabajo y se mencionan posibles soluciones:

Es necesaria la incorporación de algoritmos de detección de movimiento para combatir el problema que se presenta cuando el objetivo a seguir es tan rápido que el sistema pierde su ubicación. Así se tendría un sistema que, apenas se perdiera el objetivo, sería capaz de ubicar las zonas de la imagen en las que hay movimiento para aplicar en ellas la medida de correlación y volver a encontrar el objeto de interés.

Es importante también la inclusión de filtros, como el Kalman, que ayudarían al tratamiento adecuado de las oclusiones, pues en palabras simples, este tipo de filtros puede ser usado para "predecir" la trayectoria de un objeto basándose en las características del movimiento que ha presentado dicho objeto.

Sería deseable proporcionar al sistema libertad de movimiento para tener un rango de visión que no esté limitado por el tamaño del área que puede ser mostrada por la cámara. Esto significaría construir un sistema a base de motores que sea capaz de mover la cámara de forma horizontal (ronza) y vertical (elevación) manteniendo el objetivo al centro de la imagen. Y mejor aún sería que el sistema que proporcione el movimiento tuviera velocidad variable para poder rastrear objetos con comportamiento extraño y velocidad cambiante.

Finalmente, al usar un sistema que permita movimiento a la cámara, sería necesaria la incorporación de un sistema estabilizador, para que el movimiento de los motores y la cámara no afecten el desempeño del proceso de seguimiento.

Apéndice A

Análisis de distribuciones GNU/Linux

La distribución Linux a elegir debe contar con las siguientes características:

- Facilidad de instalación.
- Facilidad de uso y personalización.
- Buena documentación.

Una buena fuente de información que ayudó a tomar tal decisión fue la lista "El top 10 de distribuciones" publicada en la pagina "DistroWatch" [13], la cual está basada en los resultados de encuestas a usuarios Linux, las cuales arrojan un estimado confiable de las distribuciones Linux más ampliamente utilizadas en todo el mundo.

Los resultados publicados en DistroWatch el día 20 de Junio de 2008 son los siguientes, empezando por la distribución más popular:

1. Ubuntu.

- **Pros:** Cuenta con actualizaciones periódicas y muy buen soporte técnico; amigable para el usuario reciente, cuenta con abundante documentación, recibe contribuciones oficiales y de usuarios. Al estar basada en Debian hereda de esta una muy buena estabilidad.
- **Contras:** Algunos de los paquetes Ubuntu como Launchpad o Rosetta poseen marca registrada y carecen de compatibilidad con Debian.
- **Manejo de paquetes de software:** Herramientas avanzadas de manejo de paquetes (APT) al usar paquetes de Debian.
- **Ediciones disponibles:** Ubuntu, Kubuntu, Edubuntu, Xubuntu, UbuntuStudio y MythUbuntu para procesadores de 32-bit (i386) y 64-bit (x86_64); Ubuntu cuenta con ediciones para servidores con procesadores SPARC.

2. openSUSE

- **Pros:** Cuenta con herramientas de configuración intuitivas y seguras; numerosos paquetes de repositorios y software y, un sitio web con excelente infraestructura y vastas publicaciones.

- **Contras:** El convenio establecido entre Novell y Microsoft en noviembre de 2006 el cual aparentemente legaliza los derechos de propiedad intelectual de Microsoft sobre Linux; la instalación de su pesado escritorio y las aplicaciones gráficas es a veces visto como "saturado y lento".

- **Manejo de paquetes de software:** Aplicación gráfica y línea de comando YaST para paquetes RPM.

- **Ediciones disponibles:** openSUSE para 32-bit (i386), 64-bit (x86_64) y procesadores PowerPC (ppc) (también cuenta con una edición DVD Live no instalable); Empresa SUSE Linux Enterprise Escritorio/Servidores para arquitecturas i586, IA64, PowerPC, s390, s390x y x86_64.

3. Fedora.

- **Pros:** Altamente innovador; sorprendentes características de seguridad; un gran número de paquetes respaldados y una estricta adherencia a la filosofía del software libre.

- **Contras:** Al estar auspiciada por Redhat, sus prioridades tienden a respaldar características empresariales y de pruebas más que a las necesidades de escritorio de los usuarios.

- **Manejo de paquetes de software:** Aplicación de línea de comandos gráfico YUM de paquetes RPM.

- **Ediciones disponibles**: Fedora para procesadores de 32-bit (i386), 64-bit (x86_64) y PowerPC (ppc); Red Hat Linux Empresa para arquitecturas i386, IA64, PowerPC, s390x y x86_64; cuenta también con live CD y ediciones en live DVD.

4. Debian GNU/Linux.

- **Pros:** Muy estable y personalizable; destacado control de calidad; incluye más de 20,000 paquetes de software; soporta más arquitecturas de procesador que cualquier otra distribución de Linux.
- **Contras:** Conservadora, ya que al dar soporte a varias arquitecturas de procesador, no siempre incluye las tecnologías más nuevas; ciclo de lanzamientos lento (un lanzamiento estable cada 1 - 3 años); las discusiones entre desarrolladores en las listas de correo devienen groseras de vez en cuando; su instalación y personalización requiere de experiencia en entornos Linux.
- **Manejo de paquetes de software:** Advanced Package Tool (APT) usando paquetes DEB.
- **Ediciones disponibles:** CD/DVD de instalación e imágenes de live CD para 11 arquitecturas de procesador, incluyendo procesadores Power, AMD e Intel de 32-bit y 64-bit, entre otras.

5. Mandriva Linux.

- **Pros:** Amigable para los principiantes, especialmente en sus versiones comerciales; excelente utilidad de configuración central; muy buen soporte nativo para docenas de lenguajes; live CD instalable.
- **Contras:** El Servicio al cliente de la compañía ha adquirido una mala reputación en los últimos años; la infraestructura de su sitio web es compleja y confusa; popularidad decreciente debido a su naturaleza comercial y a algunas decisiones corporativas impopulares.
- **Manejo de paquetes de software:** URPMI con Rpmdrake (una interfaz gráfica para URPMI) usando paquetes RPM; dispone de "SMART" como método alternativo.
- **Ediciones disponibles:** Mandriva ediciones Free y One de descarga libre, para procesadores 32-bit (i386) y 64-bit (x86_64); ediciones comerciales Mandriva Discovery, PowerPack y PowerPack Plus para procesadores 32-bit (i386) y 64-bit (x86_64); también posee soluciones corporativas de alto nivel orientadas al escritorio, servidores y firewalls, con opción de soporte de largo alcance.

6. Linux Mint.

- **Pros:** Gran cantidad de paquetes disponibles; cientos de aplicaciones de mejora de la interfaz y accesibilidad; inclusión de códecs multimedia; abierta a las recomendaciones de sus usuarios.
- **Contras:** Las ediciones alternativas "community" no siempre incluyen las últimas versiones de sus paquetes. El proyecto no contempla un sistema de seguimiento de *bugs* y seguridad.
- **Manejo de paquetes de software:** APT con mintInstall usando paquetes .DEB (compatible con repositorios Ubuntu).
- **Ediciones disponibles:** Una edición principal (con GNOME) para computadoras con procesadores de 32 y 64 bits, además de una variedad de ediciones "community" para procesadores de 32 bits (con KDE, Xfce y Fluxbox).

7. PCLinuxOS

- **Pros:** Soporte nativo para controladores gráficos, plugins de navegador y códecs multimedia; inicio rápido; software actualizado.
- **Contras:** No ofrece edición de 64-bit; sólo tiene soporte nativo para el idioma inglés; carece de plan de lanzamientos.
- **Manejo de paquetes de software:** Advanced Package Tool (APT) usando paquetes RPM.
- **Ediciones disponibles:** Ediciones MiniMe, Junior y BigDaddy para arquitecturas de procesador de 32-bit (i586).

8. Slackware Linux.

- **Pros:** Muy estable; limpia y libre de errores; fuerte adherencia a los principios UNIX.
- **Contras:** Selección limitada de aplicaciones oficialmente soportadas; conservativa en términos de la selección básica de paquetes; procedimiento de actualización complejo; no existe una versión oficial de 64bits.
- **Manejo de paquetes de software:** "pkgtools" usando paquetes TGZ (TAR.GZ).
- **Ediciones disponibles:** CD y DVD de instalación para procesadores de 32bits (i486).

9. Gentoo Linux.

- **Pros:** Excelente manejo de infraestructura, personalización y opciones de ajuste sin paralelo, lo que la hacen una distribución muy rápida en desempeño ya que cada instalación está puesta a punto para la computadora en la que se instala; superior documentación en línea.
- **Contras:** Inestabilidad ocasional y riesgo de XXX; el proyecto sufre de falta de dirección y frecuentes luchas entre los desarrolladores.
- **Manejo de paquetes de software:** "Portage" usando paquetes fuente (SRC)
- **Ediciones disponibles:** CD de instalación y "Live-CD" (con GNOME) para procesadores Alpha, AMD64, HPPA, IA64, MIPS, PPC, SPARC y x86; también existen "stages" para instalación manual desde la línea de comando

10. CentOS Linux.

- **Pros:** Versiones estables; descarga gratis y libertad de uso; sus versiones cuentan con 5 años de actualizaciones de seguridad.
- **Contras:** Entra en conflicto con las últimas tecnologías Linux; las nuevas versiones de la distribución no vienen con las últimas versiones de sus paquetes.
- **Manejo de paquetes:** Utilidad gráfica y de comandos YUM, usando paquetes RPM.
- **Ediciones disponibles:** DVD de instalación y liveCD instalable (con GNOME) para procesadores i386 y x86_64. Versiones antiguas (3.x y 4.x) también disponibles para procesadores Alpha, IA64, e IBM de la serie Z (s-390, s-390x).

Al analizar la lista anterior podemos descartar algunas distribuciones fácilmente:

OpenSUSE, ya que su estructura interna es tan grande que es considerado como "saturado y lento".

Mandriva Linux, ya que las versiones libres son muy limitadas en cuanto a disponibilidad de paquetes.

PCLinuxOS y **CentOS** son descartados por su falta de soporte y liberación de versiones muy lenta.

LinuxMint: ya que el proyecto no tiene un buen tratamiento de bugs y seguridad.

Slackware Linux no es una buena opción dado que tiene una gama limitada de paquetes oficialmente soportados.

Lo que nos deja solo 4 opciones, a saber, Gentoo, Debian GNU/Linux, Fedora y Ubuntu, las cuales se pueden categorizar en dos grupos:

- Por nivel de personalización y desempeño (Gentoo y Debian)
- Por popularidad y facilidad de instalación y uso. (Ubuntu y Fedora)

La elección final es difícil, ya que las cuatro son distribuciones muy confiables y populares. El criterio usado para elegir la distribución sobre la cual se va a trabajar es sin duda el aspecto más importante que deberá tener el entorno final, se requiere velocidad de respuesta del sistema, por lo que se tiene que elegir una distribución que permita elegir en detalle las características de desempeño del sistema, que se va a instalar y que no, para así tener el sistema lo más ligero posible.

De las 4 opciones anteriores, Gentoo y Debian reúnen esas características, en cambio Ubuntu y Fedora son mas automatizadas en el proceso de instalación y aunque solo instalan un sistema básico y dan la opción de eliminar los paquetes/aplicaciones innecesarios, sería deseable tener esa posibilidad desde el proceso de instalación.

De entre Gentoo y Debian la que mas nivel de personalización para lograr un mayor desempeño tiene es Gentoo, pues en el proceso de instalación se ajustan varios parámetros que hacen que la instalación esté en función del tipo de procesador usado, además se ajustan los parámetros de compilación para que no se creen paquetes binarios con dependencias innecesarias, aspecto del cual carece Debian.

Así pues, se llegó a la decisión de usar Gentoo Linux como plataforma para el sistema de visión que se va a realizar.

Apéndice B

Análisis y elección del Entorno Integral de Desarrollo (IDE) usado

En el mundo del software libre existe una gran variedad de herramientas para el desarrollo de software, como editores, compiladores, intérpretes, depuradores, etc. para distintos lenguajes de programación. Existen también entornos "todo en uno" en las cuales se incluyen las herramientas antes mencionadas y se ponen al alcance del programador sin necesidad de aplicaciones adicionales, este conjunto de herramientas es conocido como "Entorno Integral de Desarrollo" (IDE por sus siglas en inglés).

Revisando la bibliografía y los foros de programación se puede encontrar que las IDE's más potentes y populares en entornos Linux son:

- Eclipse
- Netbeans
- KDevelop
- Anjuta

A continuación se mencionan las características de cada uno de ellos.

Eclipse.[14]

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma que emplea módulos (en inglés *plug-in*) para proporcionar toda su funcionalidad, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Este mecanismo de módulos es una plataforma ligera para componentes de software. Eclipse puede extenderse usando otros lenguajes de programación como son C/C++ y Python o trabajar con lenguajes para procesado de texto como LaTeX, aplicaciones en red como Telnet y Sistema de gestión de base de datos.

La versión actual de Eclipse dispone de las siguientes características:

- Editor de texto
- Resaltado de sintaxis
- Compilación en tiempo real
- Pruebas unitarias con JUnit
- Control de versiones con CVS
- Integración con Ant
- Asistentes (*wizards*): para creación de proyectos, clases, tests, etc.
- Refactorización

Asimismo, a través de "plugins" libremente disponibles es posible añadir:

- Control de versiones con Subversion.
- Integración con Hibernate.

Eclipse fue liberado originalmente bajo la Common Public License, pero después fue re-licenciado bajo la Eclipse Public License. La Free Software Foundation ha dicho que ambas licencias son licencias de software libre, pero son incompatibles con Licencia Pública General de GNU (GNU GPL). Mike Milinkovich, de la fundación Eclipse comentó que el cambio a la GPL será considerado cuando la versión 3 de la GPL sea liberada

A continuación una figura que ilustra la pantalla principal del entorno IDE.

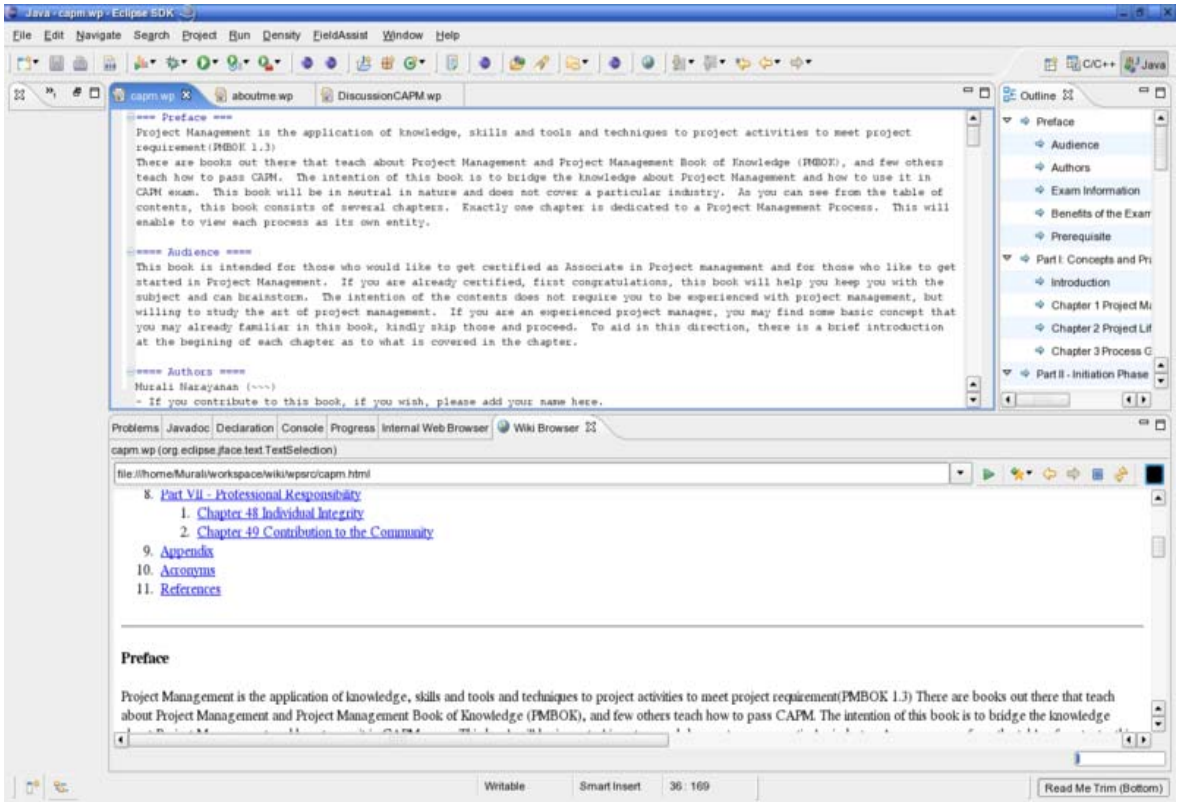


Figura B.1 Eclipse IDE.

NetBeans.[15]

NetBeans se refiere a una plataforma para el desarrollo de aplicaciones de escritorio usando Java y a un entorno de desarrollo integrado (IDE) desarrollado usando la Plataforma NetBeans.

La Plataforma NetBeans es una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio grandes. La plataforma ofrece servicios comunes a las aplicaciones de escritorio, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación. Entre las características de la plataforma están:

- Administración de las interfaces de usuario (ej. menús y barras de herramientas) con código Java
- Administración de las configuraciones del usuario
- Administración del almacenamiento (guardando y cargando cualquier tipo de dato)
- Administración de ventanas
- Framework basado en asistentes (diálogos paso a paso)

El IDE NetBeans es un IDE - una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java - pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans. El IDE NetBeans es un producto libre y gratuito sin restricciones de uso.

El NetBeans IDE soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). Entre sus características se encuentra un sistema de proyectos basado en Ant, control de versiones y refactoring.

La versión actual es NetBeans IDE 6.1, la cual fue lanzada el 28 de Abril de 2008. NetBeans IDE 6.1 extiende las características existentes del Java EE (incluyendo Soporte a Persistencia, EJB 3 y JAX-WS). Adicionalmente, el NetBeans Enterprise Pack soporta el desarrollo de Aplicaciones empresariales con Java EE 5, incluyendo herramientas de desarrollo visuales de SOA, herramientas de esquemas XML, orientación a web servicies (for BPEL), y modelado UML. El NetBeans C/C++ Pack soporta proyectos de C/C++.

Desde Julio de 2006, NetBeans IDE es licenciado bajo la Common Development and Distribution License (CDDL), una licencia basada en la Mozilla Public License (MPL).

En la siguiente figura se muestra una captura de pantalla del IDE Netbeans.

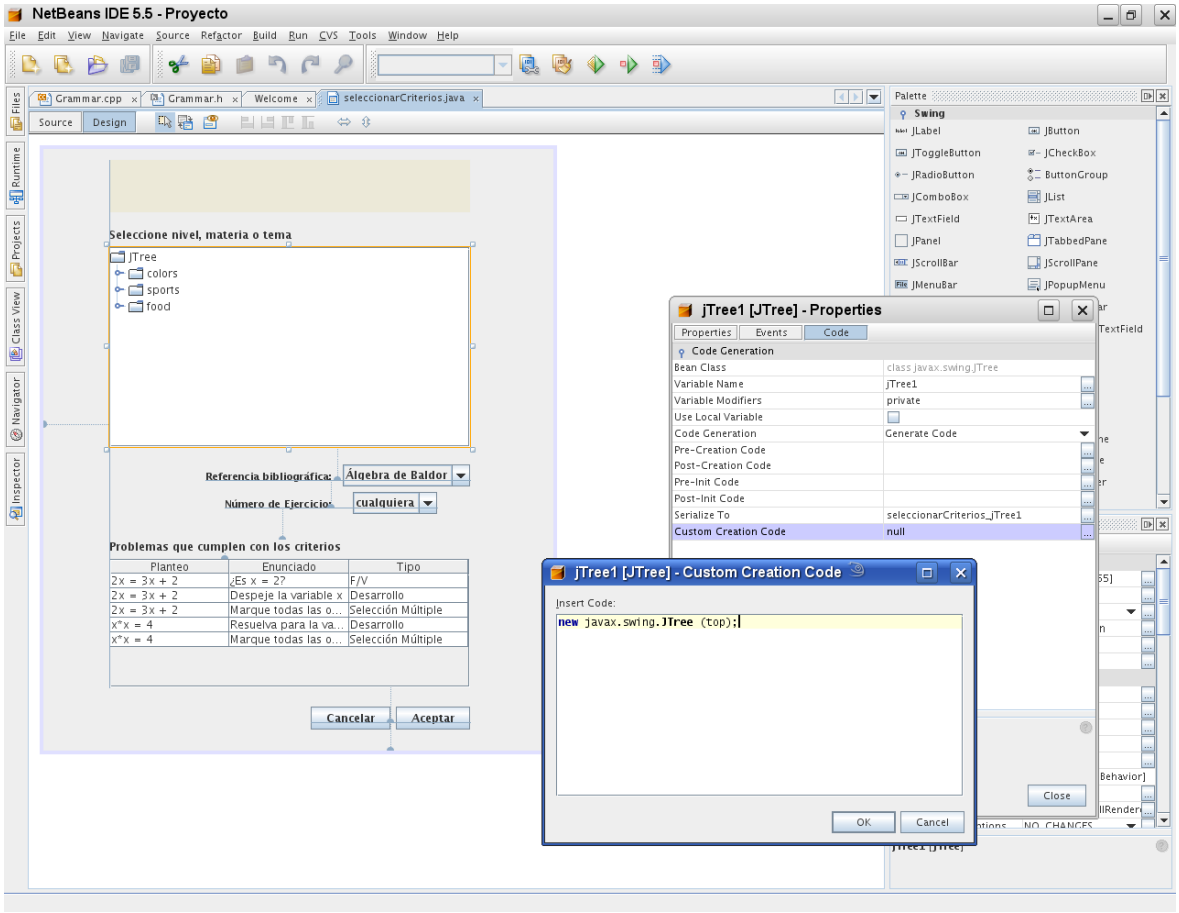


Figura B.2 Netbeans IDE.

KDevelop.[16]

KDevelop es un entorno de desarrollo integrado para sistemas Linux y otros sistemas Unix, publicado bajo licencia GPL, orientado al uso bajo el entorno gráfico KDE, aunque también funciona con otros entornos, como Gnome.

El mismo nombre alude a su perfil: KDevelop - KDE Development Environment (Entorno de Desarrollo para KDE).

KDevelop 3.0 ha sido reconstruido completamente desde los cimientos, se dio a conocer junto con KDE 3.2 en diciembre de 2003.

A diferencia de muchas otras interfaces de desarrollo, KDevelop no cuenta con un compilador propio, por lo que depende de gcc para producir código binario.

Su última versión se encuentra actualmente bajo desarrollo y funciona completamente con distintos lenguajes de programación como C, C++, Java, Ada, SQL, Python, Perl y Pascal, así como guiones para el intérprete de comandos Bash.

KDevelop usa por defecto el editor de texto Kate. Las características que se mencionan a continuación son específicas del entorno de desarrollo:

- Editor de código fuente con destacado de sintaxis e indentado automático (Kate).
- Gestión de diferentes tipos de proyectos, como *Automake*, *qmake* (para proyectos basados en la biblioteca Qt y *Ant* (para proyectos basados en Java).
- Editor de interfaces usando la biblioteca Qt, a partir de las cuales se genera código en C++.
- Navegador entre clases de la aplicación.
- Front-end para gcc, el conjunto de compiladores de GNU.
- Front-end para el depurador de GNU.
- Asistentes para generar y actualizar las definiciones de las clases y el framework de la aplicación.
- Completado automático del código en C y C++.

- Compatibilidad nativa con Doxygen.
- Permite control de versiones.

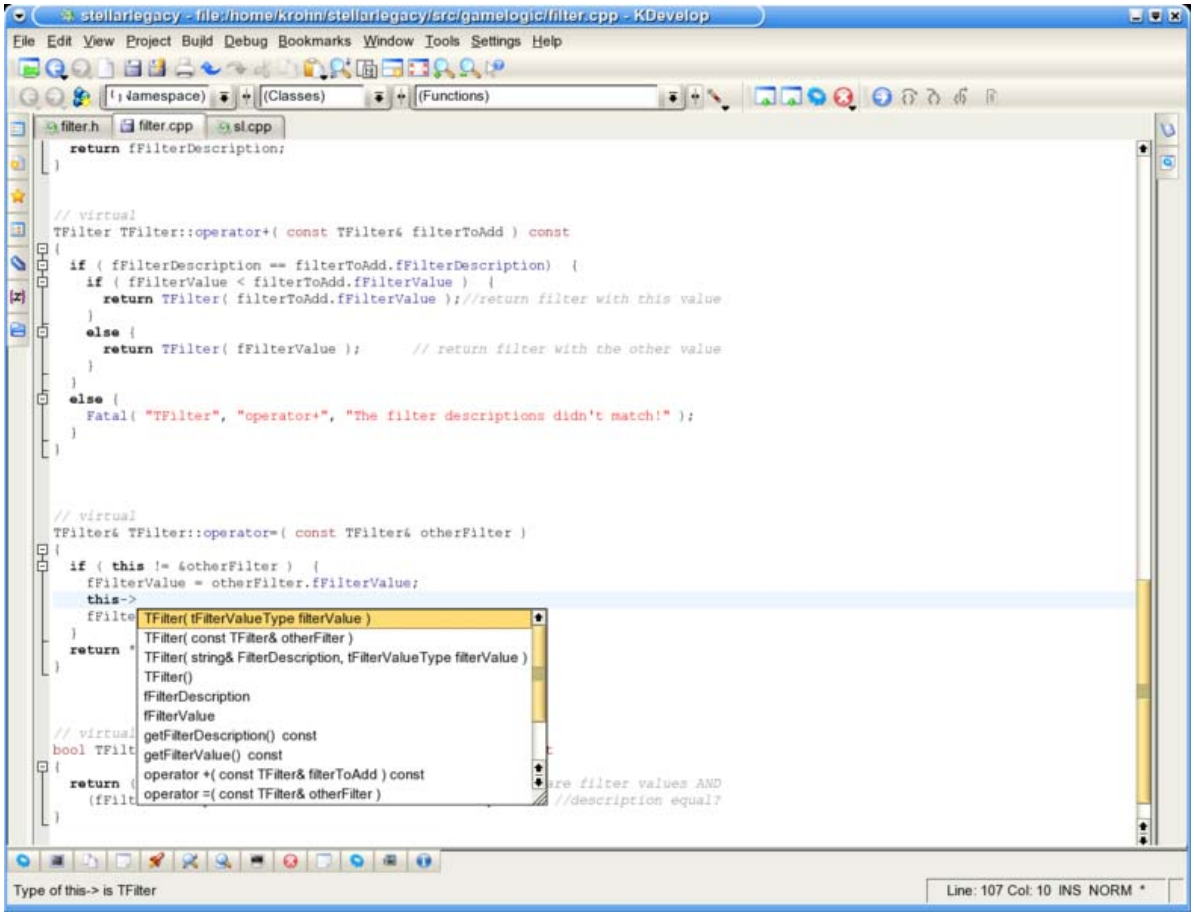


Figura B.3 KDevelop IDE.

Anjuta.[17]

Anjuta (actualmente Anjuta DevStudio) es un entorno integrado de desarrollo (IDE) para programar en los lenguajes C, C++, Java y Python, en sistemas GNU/Linux. Su principal objetivo es trabajar con GTK y en el *escritorio* GNOME, además ofrece un gran número de características avanzadas de programación. Anjuta es software libre, liberado bajo la licencia GPL.

Incluye un administrador de proyectos, asistentes, plantillas, depurador interactivo y un poderoso editor que verifica y resalta la sintaxis escrita.

Actualmente, aunque se continúa el mantenimiento de la última versión estable, Anjuta 1.2, la versión 2 tiene importantes mejoras entre las que destaca:

- nuevo sistema de extensiones, todos los de la primera versión son compatibles.
- arquitectura revisada y extensible.
- nuevo Intérprete de comandos propio y documentación del API.
- integrado un nuevo sistema de ayuda.
- un diseñador gráfico de interfaces de usuario (todavía incompleto) con Glade.
- mejoras diversas en el editor de programación (edición remota, mejor realce de sintaxis, etc.).
- nuevo administrador de tareas.
- extensión para añadir macros, insertar texto predefinido o personalizado.
- plantilla fácilmente extensible para proyectos mediante asistente.
- extensión para Subversion (todavía incompleto)
- administrador de sesiones de trabajo.
- actualizado la extensión para CVS.

A continuación se muestra una captura del entorno integral de desarrollo Anjuta.

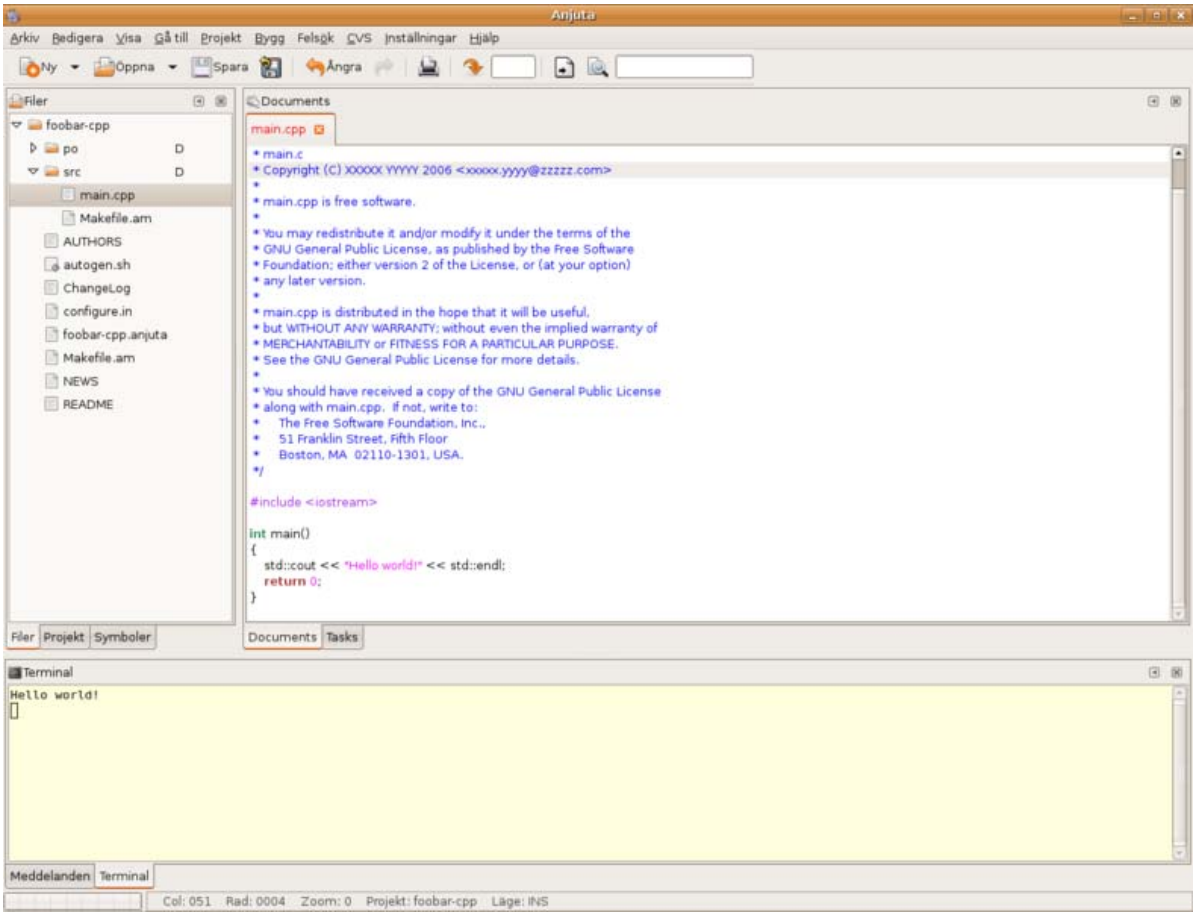


Figura B.4 Anjuta IDE.

Elección:

Para determinar el IDE a usar en el desarrollo de la aplicación de esta tesis es necesario que dicho entorno permita usar C (ó bien C++) como lenguaje de programación (pues es el lenguaje con el que más experiencia de programación se tiene) y que permita crear interfaces de manera rápida, que puedan ser manipuladas mediante el mismo lenguaje.

El enfoque modular de Eclipse lo hacen muy atractivo, pero el inconveniente que se le puede encontrar para este proyecto es que el módulo que tiene para la creación de interfaces solo está disponible para usarse con código Java. Lo mismo sucede con NetBeans.

El diseñador de interfaces de Anjuta si permite el uso de código C/C++ para manipular los componentes de la interfaz (botones, etiquetas, controles, etc.) pero aun está en fase de desarrollo por lo que está incompleto y aún es inestable.

KDevelop, es una buena opción, pues tiene integrado por completo un diseñador de interfaces llamado diseñador de KDE (basado en QtDesigner), el cual crea archivos a partir de los cuales se genera código en varios lenguajes, incluido C/C++, lenguaje para el que funcionan correctamente el editor, compilador, depurador (*gdb*), etc. A pesar de estar pensado para KDE, KDevelop funciona perfectamente con Gnome.

Como resultado de este pequeño análisis se tomó la decisión de usar KDevelop, pues integra todas las herramientas que necesita un programador.

Apéndice C

Características de la webcam utilizada

La QuickCam Pro 9000 de la marca Logitech es una de las webcams mas aventajadas en la relación calidad/prestaciones/precio. Está diseñada para funcionar con los sistemas operativos Microsoft Windows XP y Windows Vista, pues el software de filtros y efectos que la acompañan solo sirve bajo estos sistemas, sin embargo, funciona correctamente en sistemas operativos GNU/Linux, ya que cumple con el estándar UVC (USB Video Class) el cual le proporciona los drivers necesarios para dicho propósito a nivel kernel usando *V4L2*.

Los requisitos para poder usarla en entornos Windows son:

Windows XP

- Procesador Pentium IV (o compatible), 1,4 GHz (se recomiendan 2,4 GHz)
- 128 MB de RAM (se recomiendan 256 MB)
- Disco duro con 200 MB de espacio libre
- Adaptador de vídeo en color de 16 bits
- Altavoces y tarjeta de sonido compatible con Windows (se recomienda tarjeta de sonido con dúplex completo)
- Puerto USB
- Unidad de CD-ROM

Windows Vista

- Procesador Pentium IV (o compatible), 2,4 GHz (se recomiendan 2,8 GHz)
- 512 MB de RAM (se recomienda 1 GB)
- Disco duro con 200 MB de espacio libre
- Adaptador de vídeo en color de 16 bits
- Altavoces y tarjeta de sonido compatible con Windows (se recomienda tarjeta de sonido con dúplex completo)
- Puerto USB
- Unidad de CD-ROM

Y aunque esos requisitos sean para Windows, nos dan una idea de lo necesario para poder usarla en sistemas GNU/Linux.

Las especificaciones técnicas de la webcam son:

- Lente Carl Zeiss
- Sistema de enfoque automático
- Sensor de 2 megapíxeles de resolución superior con tecnología RightLight2
- Profundidad de color: Color verdadero de 24 bits
- Captura de vídeo: Hasta 1600 x 1200 píxeles (calidad HD)
- Frecuencia de cuadro: Hasta 30 fps
- Captura de fotografías: 8 megapíxeles (mejora por software)
- Micrófono integrado con tecnología RightSound

Figura C.1 Webcam QuickCam Pro 9000 de Logitech

Apéndice D

Código fuente de la aplicación

Con el fin de no hacer más extenso este documento se ha decidido no incluir el código completo de la aplicación y en su lugar solo se muestra el código de las funciones más importantes para el funcionamiento del sistema, las cuales se acompañan de una breve explicación. El código completo de la aplicación se encuentra en el LiveDVD de la aplicación, en una carpeta llamada "src" que está contenida a su vez por la carpeta "*trackeador*" ubicada en el escritorio.

A continuación se muestra el código encargado de llenar la matriz correspondiente a la imagen original, tomando como fuente una imagen capturada.

```
void seguimiento::llenarMatrizOriginal( unsigned char *imagePixelPointer )
{
   //promedio ponderado
   float alpha, beta, gamma;
   alpha=0.2125;
   beta=0.7154;
   gamma=0.0721;

   unsigned char *indice;
   int R, G, B;

   for(int i=0; i<imagenAlto; i++)
      for(int j=0; j<imagenAncho; j++)
      {
         //se multiplica a j por 4 para contemplar imágenes de 4 canales
         indice = imagePixelPointer + (i* bytesPorLinea) + (j * 4);
         B = (*indice); indice++;
         G = (*indice); indice++;
         R = (*indice);
         matrizImagen[i][j] = (unsigned char)(alpha*R + beta*G + gamma*B);
      }
}
```

Como se observa en el código anterior, la función recibe como parámetro un apuntador a la región de memoria en donde se encuentra almacenada la información de los pixeles de la imagen capturada. El proceso de llenado se hace mediante un ciclo que se mantiene vivo mientras haya columnas y renglones por recorrer en la imagen. Dentro del ciclo se usa un polinomio de direccionamiento que contempla renglones y columnas para ubicar la posición de cada uno de los pixeles de la imagen, posición que se almacena en una variable índice, a partir de la cual se obtienen los valores de los canales R, G y B, con los que se realiza un promedio ponderado que deja un solo valor en escala de grises para el pixel en cuestión. Finalmente se almacena este valor dentro de una matriz en la posición cartesiana que le corresponda dentro de la imagen.

Algo análogo sucede con el código que extrae los pixeles de la subimagen patrón y llena con ellos la matriz correspondiente:

```
void seguimiento::llenarMatrizPatron( int posX, int posY, unsigned char
*imagePixelPointer)
{
   //promedio ponderado
   float alpha, beta, gamma;
   alpha=0.2125;
   beta=0.7154;
   gamma=0.0721;

   unsigned char *indice;
   int R, G, B;
   int doblePatron = radioPatron*2;

   for(int i=0; i<doblePatron; i++)
    for(int j=0; j<doblePatron; j++)
    {
       indice = imagePixelPointer + ((i+posY)*bytesPorLinea) + ((j+posX)*4);
       B = (*indice); indice++;
       G = (*indice); indice++;
       R = (*indice);
       matrizImagenPatron[i][j] = (unsigned char)(alpha*R + beta*G + gamma*B);
    }
}
```

La diferencia es que para llenar la matriz de la región patrón es necesario contemplar la posición (x,y) del objeto de interés dentro de la imagen, pues solo se almacenarán los pixeles que se encuentran alrededor del punto del "clic" y que no estén fuera de la zona patrón que es determinada por el valor del radio establecido para dicha zona.

Para realizar el seguimiento de objetos se usa el siguiente hilo de ejecución:

```
void hilo::run()
{
  while(capturar)
  {
    //captura de una imagen de la webcam, la imagen queda en "pixelImagen"
    wcam_read_frame_rgb(cam_fd, pixelImagen);

    if (enganchado)
    { //el usuario ha dado clic en un objetivo

      if(primeraVez)
      {  //en el primer ciclo se tiene que crear el objeto que realizará
         //el seguimiento y se tiene que llenar la matriz patrón

         validarCoordenadas();  //se valida el clic del usuario

         trackeador = new seguimiento(imageAlto, imageAncho, rowstride,
                                      radioBusqueda, radioPatron);

         trackeador->llenarMatrizPatron(coordMouseX-radioPatron,
                                        coordMouseY-radioPatron,  pixelImagen);

         primeraVez = false; //los siguientes ciclos ya no entrará aquí
      }

      else
      { //no es la primera vez que entra al ciclo

        //se realiza el seguimiento:
        trackeador->trackeando(coordMouseX, coordMouseY, pixelImagen);

        //se actualizan las coordenadas con la nueva posición
        coordMouseX = trackeador->obtenPosX();
        coordMouseY = trackeador->obtenPosY();

        if (!enganchado)
          delete trackeador;

        validarCoordenadas();

        //Aquí va la parte de código que dibuja los cuadros patrón y búsqueda
        //que sirven de mira para señalar el objetivo.
      }
    }

    if(actualizar)
    {
      //Aquí va el código para el repintado de la imagen sobre el lienzo
    }

  }
}
```

El proceso consiste en un ciclo que está activo mientras no se presione un botón que termina la captura de imágenes. Para que se inicie el seguimiento es necesario que el usuario haya dado "clic" en un objetivo, con lo que la bandera *enganchado* obtendría el valor *true*, dando paso a una secuencia de acciones que solo se realizan en el primer ciclo: se crea el objeto que realizará el seguimiento y se llena la matriz de pixeles de la región patrón (estas acciones se realizan de nuevo cuando el usuario selecciona un nuevo objetivo).

En los ciclos posteriores se invoca a la función *trackeando* la cual se encarga de realizar el seguimiento propiamente dicho, después se actualizan las coordenadas en donde se encontró el mejor candidato a objetivo y finalmente se actualiza la imagen mostrada.

El cuerpo de la función *trackeando* es el siguiente:

```
void seguimiento::trackeando(int coordMouseX, int coordMouseY,
                             unsigned char *imagePixelPointer)
{
   llenarMatrizOriginal(imagePixelPointer);

   obtenerCandidatos(coordMouseX, coordMouseY);

   //Se resta "radioPatron" para indicar que se trata la posición superior
   //izquierda de la ventana de búsqueda
   llenarMatrizPatron(posX-radioPatron, posY-radioPatron, imagePixelPointer);
}
```

Lo primero que se hace en dicha función es llenar la matriz que contiene los pixeles de la imagen completa, después se invoca la función *obtenerCandidatos*, dentro de la cual se realiza la correlación que deja la nueva coordenada del objetivo en *posX* y *posY*, posteriormente se actualiza la matriz patrón con los pixeles de la nueva posición del objetivo para que se utilice en ciclos posteriores.

La función *obtenerCandidatos*, como su nombre lo indica, se encarga de generar todas las ventanas candidato que caben dentro de la zona de búsqueda y aplicarles la correlación *CMS*. Los valores obtenidos por la función de similaridad se almacenan en una estructura que contiene una coordenada y un valor de similitud, para que después se haga una búsqueda de la correlación más alta de todas, la cual deberá corresponder a la zona candidato que más se asemeje al objeto de interés, es decir, se conocerá la nueva posición del objetivo por medio de las coordenadas *x* e *y* almacenadas. El cuerpo de *obtenerCandidatos* es el siguiente:

```
void seguimiento::obtenerCandidatos(int posMouseX, int posMouseY)
{
   int tam = ((2*radioPatron)+1) * ((2*radioPatron)+1);

   VALORES_CORRELACION *valoresCorrelacion;
   valoresCorrelacion = new VALORES_CORRELACION[tam];

   int x1, y1, x2, y2;
   int pos=0;
   double val;
   double max = -1;


   //a partir de la posición del objetivo se obtienen las coordenadas
   //(x1,y1) y (x2,y2) de la ventana de búsqueda
   x1 = posMouseX - 2*radioPatron;
   x2 = x1+2*radioPatron;
   y1 = posMouseY - 2*radioPatron;
   y2 = y1+2*radioPatron;

   //se aplica la correlación a todas las ventanas candidatas
   for(int i=y1; i<=y2; i++)
   {
      for(int j=x1; j<=x2; j++)
      {
         val = correlacionCMS( matrizImagenPatron, matrizImagen, i, j,
                               radioPatron);

         valoresCorrelacion[(2*radioPatron+1)*(i-y1) + j-x1].valor = val;
         valoresCorrelacion[(2*radioPatron+1)*(i-y1) + j-x1].x = j;
         valoresCorrelacion[(2*radioPatron+1)*(i-y1) + j-x1].y = i;
      }
   }

   //se obtiene la correlación más alta de todas
   for(int i=0; i< tam; i++)
   {
      if (valoresCorrelacion[i].valor > max)
      {
         max = valoresCorrelacion[i].valor;
         pos = i;
      }
   }

   //se actualizan las variables que indican la posición del objetivo.
   //se les suma “radioPatron” para que correspondan al centro de la ventana
   //patrón en vez de la esquina superior izquierda
   posX = valoresCorrelacion[pos].x + radioPatron;
   posY = valoresCorrelacion[pos].y + radioPatron;

   cout<<endl<<"Nuevo centroide = "<<valoresCorrelacion[pos].x<<","
       <<valoresCorrelacion[pos].y<<")"<<endl;

   cout<<"Indice de similitud = "<<valoresCorrelacion[pos].valor<<endl;

   delete[] valoresCorrelacion;
}
```

Para finalizar, es necesario mostrar el código que realiza la función de correlación, la cual fue explicada en el capítulo 3 en la estrategia de solución general y es fácilmente deducible de la formula de la *CMS*. El código es el siguiente:

```
double seguimiento::correlacionCMS (unsigned char **matPatron,
                            unsigned char **matImagen, int a, int b, int radio)
{
  double media_subimagen_patron=0, media_subimagen_candidato=0;
  double valor=0;
  int MxN;
  double matPat_menos_mediaPatron=0, matCand_menos_mediaCand=0;
  double sumatoriaSuperior=0, sumatoriaInferior=0;

  MxN = ((2*radio)+1)*((2*radio)+1);

  for(int i=0; i<=2*radio; i++)
  {
    for(int j=0; j<=2*radio; j++)
    {
      media_subimagen_patron += (double)matPatron[i][j];
      media_subimagen_candidato += (double)matImagen[i+a][j+b];
    }
  }

  media_subimagen_patron /= (double)MxN;
  media_subimagen_candidato /= (double)MxN;


  double aux1=0,aux2=0;
  for(int i=0; i<=2*radio; i++)
  {
   for(int j=0; j<=2*radio; j++)
    {
     matPat_menos_mediaPatron = (double)matPatron[i][j]-media_subimagen_patron;
     matCand_menos_mediaCand = (double)matImagen[i+a][j+b] - media_subimagen_candidato;
     sumatoriaSuperior += matPat_menos_mediaPatron * matCand_menos_mediaCand;
     aux1 += matPat_menos_mediaPatron * matPat_menos_mediaPatron;
     aux2 += matCand_menos_mediaCand * matCand_menos_mediaCand;
    }
  }

  sumatoriaInferior = sqrt(aux1*aux2);
  valor = sumatoriaSuperior / sumatoriaInferior;

  return valor;
}
```

Con esto se concluye el código de las funciones mas relevantes de la aplicación.

Bibliografía

[1] http://es.wikipedia.org/wiki/Ley_de_Moore, “Ley de Moore”.

[2] S. A. Motamedi A. Berhad, A. Shahrokni. A robust vision-based moving target detection and tracking system. In *Proceedings of Image and Vision Computing conference*. IEEE, Noviembre 2001.

[3] M.M. Trivedi M.W. Eklund, G. Ravichandran. Real-time visual tracking using correlation techniques. In Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on, pages 256{263, Diciembre 1994.

[4] Hernández, M., Cabrera, J., Castrillón, M., y Guerra, C.: “DESEO: An Active Vision System for Detection, Tracking and Recognition”, International Conference on Vision Systems ICVS'99, enero 1999, págs. 376-291

[5] Hutchinson, S., Hager, G. D. y Corke, P. I.: "AT utorial on Visual Servo Control", IEEE Trans. Robot. Automat., vol. 12, núm. 5, págs. 651-670

[6] Hall, D. y Crowley, J. L.: "Tracking Fingers and Hands with a Rigid Contour Model in am Augmented Reality", 1st International Workshop on Managing Interactions in Smart Environments, 1999.

[7] M.M. Trivedi M.W. Eklund, G. Ravichandran. Real-time visual tracking using correlation techniques. In *Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on*, pages 256-263, Diciembre 1994

[8] U. von Seelen and R. Bajcsy. Adaptive correlation tracking of targets with changing scale. In Technical report, GRASP Laboratory Technical Report, June 1996

[9] http://www.linuxtv.org/downloads/video4linux/API/V4L2_API/

[10] http://www.remastersys.klikit-linux.com/

[11] Martínez C. J., Seguimiento de objetos utilizando visión foveal basada en un nuevo método de foveación, tesis de maestría. INAOE, 2006.

[12] Sucar, L. Procesamiento de imágenes y visión computacional.
ccc.inaoep.mx/~esucar/Vision/vis08v05-movimiento.ppt
INAOE.

[13] DistroWatch, Top Ten Distributions.
http://distrowatch.com/dwres.php?resource=major

[14] Wikipedia, Eclipse IDE.
http://es.wikipedia.org/wiki/Eclipse_(software)

[15] Wikipedia, NetBeans IDE
http://es.wikipedia.org/wiki/NetBeans_IDE

[16] Wikipedia, KDevelop IDE
http://es.wikipedia.org/wiki/KDevelop

[17] Wikipedia, Anjuta IDE
http://es.wikipedia.org/wiki/Anjuta