ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"Diseño, Análisis e Implementación de un Sistema de Configuración Automático de Parámetros de una Cámara en Soft Real Time"

INFORME DE MATERIA DE GRADUACIÓN

Previo a la obtención del Título de:

INGENIERO EN CIENCIAS COMPUTACIONALES

ESPECIALIZACIÓN EN SISTEMAS MULTIMEDIA

Presentado por:

ISABEL ABIGAIL ANDRADE VELARDE

GEOVANNY SEGUNDO SORIA ALAVA

GUAYAQUIL – ECUADOR

2013

AGRADECIMIENTO

A Dios que me permite compartir este triunfo con mis padres y hermanas que son mi bendición y motivación para seguir a lo largo de mi vida profesional y personal, a Bruno por estar aquí siempre apoyándome y dándome fuerzas para seguir, a mis amigos y profesores que me apoyaron en mi vida universitaria, y un agradecimiento especial al PhD. Daniel Ochoa y al Ing. Hector Peñafiel por la paciencia y conocimiento brindado durante la materia de graduación.

ISABEL ANDRADE.

A Dios que me permite alcanzar este logro y compartirlo con mi familia. A mis padres que han sido mi apoyo durante todo este tiempo, por los valores que me han inculcado pero sobre todo por ser el mejor ejemplo de vida a seguir. Un agradecimiento especial al PhD. Daniel Ochoa y al Ing. Hector Peñafiel que con sus conocimientos y apoyo hicieron posible este logro.

GEOVANNY SORIA.

DEDICATORIA

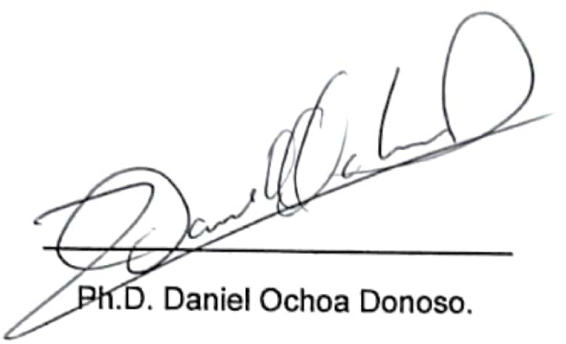
Este proyecto está dedicado a Dios y a mi familia.

ISABEL ANDRADE.

Dedico este trabajo principalmente a Dios por haberme dado la fortaleza y convicción necesaria para llegar a este momento tan importante de mi formación profesional. A mis padres que han sido el pilar fundamental en mi vida ya que con su apoyo he podido alcanzar todas las metas propuestas. A mi hermana, por el apoyo que me dio en algún instante de mi carrera.

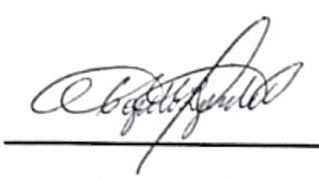
GEOVANNY SORIA.

TRIBUNAL DE SUSTENTACIÓN



Ph.D. Daniel Ochoa Donoso.

PROFESOR DE LA MATERIA DE GRADUACIÓN



Msc. Gonzalo Luzardo

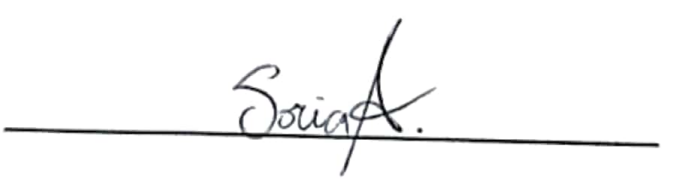
PROFESOR DELEGADO POR LA UNIDAD ACADÉMICA

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este informe de proyecto de graduación me corresponde exclusivamente; y el patrimonio intelectual de la misma a la Escuela Superior Politécnica del Litoral".

(Reglamento de exámenes y títulos profesionales de la ESPOL)

Isabel Abigail Andrade Velarde



Geovanny Segundo Soria Alava

RESUMEN

En este trabajo se presenta el análisis diseño e implementación de un sistema que permite la configuración automática de los parámetros de una cámara en soft real time.

El código del sistema está basado en la librería de V4L2 y se encuentra diseñado de tal manera que aproveche el hardware de la cámara al utilizar hilos que ejecuten sus principales tareas, además de asignarle a cada hilo de ejecución un procesador, lo cual permite disminuir tiempos de ejecución y procesamiento.

Para la configuración de los parámetros de la captura de imágenes se manipula el brillo, contraste y balance de blancos del sensor CCD de una cámara con interfaz USB.

El algoritmo utilizado en el sistema para la configuración de los parámetros se basa en la comparación del valor promedio de una imagen obtenida durante la ejecución y el valor promedio de una imagen de referencia.

ÍNDICE GENERAL

ABREVIATURAS

ESPOL	Escuela Superior Politécnica del Literal
FIEC	Facultad de Ingeniería en Electricidad y Computación
V4L	Video4Linux
RGB	Red, Green, Blue
CCD	Charge coupled device

ÍNDICE DE FIGURAS

ÍNDICE DE ECUACIONES

INTRODUCCIÓN

Actualmente con la evolución de la tecnología, el procesamiento de imágenes se ha convertido en una disciplina amplia y abarca muchos campos donde se resuelven diversos problemas usando imágenes. Algunos ejemplos son las imágenes obtenidas con fines de diagnósticos médicos, imágenes satelitales obtenidas para realizar exámenes de terreno, etc.

En la práctica, antes de poder extraer alguna información relevante de una imagen es necesario algún tipo de pre-procesamiento pues muchos algoritmos de procesamiento son susceptibles a la calidad de la imagen. Es decir que obtendremos mejores resultados de segmentación, detección de bordes, etc. si se tiene una buena imagen (imagen con brillo y contrastaste adecuado además de mostrar los colores correctos) para procesar.

Algunos de los algoritmos más utilizados en el pre-procesamiento de las imágenes son: eliminación de ruido, suavizado de imagen, realce de bordes, etc.

Si bien es cierto que los algoritmos de pre-procesamiento permiten optimizar una imagen, desarrollarlos involucra tener conocimiento en varios campos de la computación. Estos algoritmos no siempre permiten obtener los resultados esperados pues algunos no eliminan el ruido por completo y pueden implicar un aumento significativo en los tiempos de ejecución.

El pre-procesamiento podría ser evitado o reducido si un sistema es capaz de capturar imágenes con mejor calidad. Una manera de hacer esto es configurar correctamente los parámetros de la cámara. En este proyecto se abordan el análisis, diseño e implementación de un sistema que permite realizar la automatización del proceso de configuración de los parámetros de una cámara en soft real time.

En este proyecto, se obtiene una imagen de referencia que representan imágenes de “buena calidad”, luego se ajusta de forma automática los parámetros de la cámara para que las imágenes capturas tengan características similares a la imagen de referencia. El procedimiento se puede realizar de forma periódica para ajustar los parámetros a condiciones de luz cambiante

.La implementación de este proyecto emplea técnicas de programación concurrente y afinidad del CPU con el objetivo distribuir equitativamente los procesos entre los procesadores mejorando el rendimiento del procesador y reduciendo el tiempo de procesamiento. Además la captura de imágenes fue desarrollada con la librería de V4L2 que proporciona un acceso a los dispositivos de captura de video y permite configurar los parámetros de la cámara de forma dinámica.

El API de V4L2 está implementado como módulo del kernel [1] por ende hay comunicación directa entre el kernel del sistema operativo y el hardware de la cámara. En consecuencia, el API de V4L2 es más rápido ya que no presenta la sobrecarga asociada a librerías de programación más complejas. Este documento se encuentra estructurado en 6 capítulos:

En el capítulo uno se describe el análisis del problema e incluye los objetivos generales y específicos que planteamos en el presente proyecto. En el capítulo dos se presenta una breve introducción acerca del kernel del sistema operativo GNU-LINUX, algoritmos de concurrencia y afinidad de procesos. En el capítulo tres se explican conceptos básicos de calibración de cámara y el algoritmo de calibración utilizado en este proyecto. En el capítulo cuatro se explicara la arquitectura e implementación del sistema. En el capítulo cinco se explican los experimentos realizados para evaluar el

sistema desarrollado. Finalmente, en el capítulo seis se presentan los resultados y conclusiones.

CAPÍTULO 1

1 ANÁLISIS DEL PROBLEMA

En toda aplicación de visión por computadora, la calidad de la imagen resultante es un factor preponderante porque a partir de ahí se puede extraer información cualitativa de una escena y facilitar la búsqueda de información. No existe una definición objetiva de la calidad de una imagen, es más bien cuestión del juicio subjetivo del observador. Sin embargo, comúnmente los parámetros que determinan la calidad de una imagen son la resolución y la relación de contraste.

Una imagen con mala calidad requiere operaciones de pre-procesamiento para poder ser analizada de forma automática. Este pre-procesamiento se traduce en consumo de tiempo y recursos que pueden

ser reducidos si se calibran los parámetros de la cámara de tal forma que las propiedades de la imagen permitan que los algoritmos de visión por computador den mejores resultados.

Una imagen puede verse afectada por cierto factores externos que reducen la calidad de la misma, como la iluminación o el ruido electrónico en el sensor y la naturaleza de la escena. Por ejemplo si se requiere capturar a un objeto en movimiento, para una implementación eficiente, es necesario que el tiempo de pre-procesamiento sea mínimo, de lo contrario se producen pérdidas de fotogramas debido a que el procesamiento de la imagen retardaría la captura de nuevos fotogramas lo que provoca discontinuidad en la secuencia de imágenes obtenidas.

1.1 Objetivos del Proyecto

1.1.1 Objetivos Generales

El objetivo general de este proyecto es el análisis, diseño e implementación de un sistema capaz de automatizar el proceso de configuración de los parámetros de una cámara, permitiendo procesar imágenes en tiempo real, efectuando un realce selectivo de la información visual.

1.1.2 Objetivos Específicos

- Obtener una comunicación directa entre la cámara y el kernel del sistema operativo.

- Diseñar el sistema para que emplee técnicas de programación concurrente, afinidad de procesador y multiprogramación.

- Comparar los tiempos de ejecución del algoritmo de configuración de parámetros de una cámara utilizando API de V4L2 vs OPENCV**.**

1.2 Justificación

Hasta el momento se han presentado como proyectos de grado de la FIEC muchos aportes relacionados con la captura automática y procesamiento digital de imágenes para luego ser aplicados en sistemas de detección y reconocimiento de placas de vehículos, identificación de personas, etc. [2]

Nuestro proyecto se diferencia del resto por abordar dos aspectos que no habían sido considerados antes:

- Explotar los parámetros de captura para reducir el uso de algoritmos de pre-procesamiento de imágenes.

- Explotar técnicas de programación concurrente y afinidad del procesador.

En consecuencia, este sistema requiere menos recursos de hardware y software pues no depende del uso de librerías externas como OpenCV.

Este proyecto servirá como base para futuros trabajos que se vayan a desarrollar donde sea necesario explotar el kernel del sistema operativo y el hardware de la cámara.

CAPÍTULO 2

2 FUNDAMENTOS TEÓRICOS

En esta sección se explicara algunas de las técnicas y conceptos que se han empleado para el desarrollo de nuestro proyecto.

Las definiciones descritas en este capítulo se basan en el libro de P. B. G. G. G. Abraham Silberschatz, Fundamentos de Sistemas Operativos como se referencia en. [2]

2.1 Definición de Sistemas Operativos

Un sistema operativo es un programa que administra el hardware de un computador, proporcionando las bases para los programas de

aplicación y actúa como un intermediario entre el usuario y el hardware de la computadora.[2]

Los sistemas operativos funcionan en base a interrupciones, es decir si no hay ningún proceso que ejecutar este permanecerá inactivo esperando que algo ocurra, los sucesos por lo general se indican mediante la ocurrencia de una interrupción o una excepción.[2]

Interrupción: señal que envía un dispositivo de E/S a la CPU para indicar que la operación de la que se estaba ocupando, ya ha terminado.

Excepción: una situación de error detectada por la CPU mientras ejecutaba una instrucción, que requiere tratamiento por parte del SO. Para cada tipo de interrupción el sistema operativo determina que acción hay que llevar a cabo.

Para mejorar el uso de la CPU, los sistemas operativos emplean una técnica llamada multiprogramación, la cual permite tener en memoria varias tareas al mismo tiempo, asegurando que la CPU tenga siempre un trabajo que ejecutar.

2.1.1 Sistemas Multiprocesadores

La técnica de multiprocesamiento también conocido como sistemas paralelos, consiste en que varios procesadores funcionen en forma paralela compartiendo el bus del computador, la memoria y los dispositivos periféricos.

Como se menciona en la referencia[2]los sistemas multiprocesadores presentan tres ventajas:

1. **Mayor rendimiento.-** al aumentar el número de procesadores, el sistema operativo realiza más trabajo en menos tiempo esto se debe a que se puede tener en memoria varias tareas asegurando que los procesadores siempre tengan trabajo que ejecutar y de esta manera proporcionar la ilusión de que cada trabajo se está ejecutando de forma concurrente.

2. **Mayor fiabilidad.-** Obtenemos mayor fiabilidad debido a que las tareas se pueden distribuir de forma apropiada entre varios procesadores y así si falla un

procesador no hará que el sistema deje de funcionar, pues esta tarea será asignada a otro procesador.

3. **Economía de escala.-** los sistemas multiprocesador pueden resultar más baratos que tener muchos sistemas de un solo procesador, ya que pueden compartir periféricos, almacenamiento masivo y fuentes de alimentación.

2.2 Operaciones en Modo Dual

Para asegurar la correcta ejecución del sistema operativo, se debe poder distinguir entre la ejecución del sistema operativo y del código definido por el usuario. El método que usan la mayoría de los sistemas informáticos consiste en proporcionar soporte hardware que nos permite diferenciar entre el modo usuario y modo kernel[1].

Para poder diferenciar entre estos dos modos se agrega al hardware un bit denominado bit de modo, que me permite indicar el modo actual: kernel (0) o Usuario (1), con este bit se puede diferenciar entre una tarea que está siendo ejecutada en modo kernel o usuario.

[1]Núcleo de un sistema operativo, es decir, bloque de código con la parte central del funcionamiento y arranque del sistema

Modo kernel: es el espacio a memoria donde se corren los procesos del kernel, la integración con un proceso de usuario se realiza a través de llamadas al sistema.

Modo usuario: es el espacio de memoria donde corren los procesos de usuario, este espacio es protegido, el sistema previene que un proceso interfiera con otro y solo el proceso del kernel puede acceder a procesos de usuario.

Llamadas al sistema: Este proporciona la interface de llamadas al sistema que se conecta al núcleo y proporciona el mecanismo para transición entre la aplicación de modo de usuario y el kernel es decir cuando una llamada al sistema es ejecutada los argumentos de llamadas son pasadas desde el modo de usuario al modo Kernel.

2.3 Estructura de Sistemas Operativos

La ingeniería de un sistema operativo debe realizarse cuidadosamente para que el sistema funcione apropiadamente y pueda modificarse con facilidad, por lo general consisten en dividir

las tareas en componentes más pequeños, en donde cada uno de los módulos debe contener entrada, salida y funciones asociadas.

Dado que el sistema operativo tiene una gran complejidad, la modularidad es importante, por lo que se describen tres técnicas adecuadas, diseñar un sistema como una secuencia de niveles, usando un microkernel o módulos.

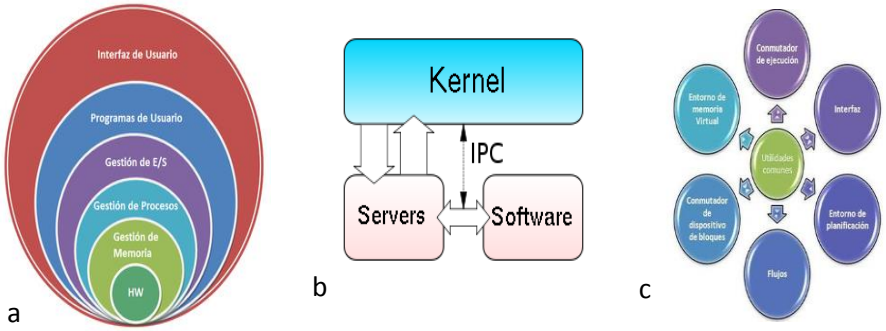


Figura 2.1 Estructura de Sistemas Operativos: a) estructura en niveles b) estructura Microkernel c) estructura Modulos

2.3.1 Estructura en Niveles

En la estructura en niveles el sistema operativo se divide en una serie de capas o niveles donde cada uno consta de estructuras de datos y rutinas y donde los niveles superiores pueden involucrar operaciones sobre los niveles inferiores.

La principal ventaja es la simplicidad de construcción y depuración, los niveles se seleccionan de modo que cada uno usa operaciones de los niveles inferiores, si durante la depuración se encuentra un error en algún nivel este tendrá que estar localizado en dicho nivel, dado que los niveles inferiores a él ya se han depurado, cada nivel oculta a los niveles superiores la existencia de determinadas estructuras de datos, operaciones y hardware.

Los sistemas operativos proporcionan una serie de servicios, en el nivel más bajo, las llamadas al sistema permiten la manipulación de archivos y dispositivos. En un nivel superior lo podemos realizar desde el intérprete de comandos o Shell permitiendo el control de programas, solicitudes de estado y solicitudes de E/S.

Una de las principales desventajas con el método de niveles es la de definir apropiadamente los diferentes niveles, dado que un nivel sólo puede usar los servicios del nivel inferior, otra desventaja importante es que tiende a ser menos eficientes que otros tipos de implementación debido una

llamada al sistema tarda más en ejecutarse que un sistema sin niveles.[2]

2.3.2 Microkernel

Al utilizar el método del microkernel se proporciona un mecanismo de comunicación entre el programa cliente y los distintos servicios que se ejecutan en el espacio de usuario.

La comunicación se realiza de forma indirecta intercambiando mensajes con el microkernel.

Una de las ventajas de este método es la facilidad para ampliar el sistema operativo, es decir todos los servicios nuevos se añaden al espacio de usuario y en consecuencia, no requiere que se modifique el kernel, el sistema operativo es más fácil de portar de un diseño de hardware a otro, también proporciona más seguridad y fiabilidad, dado que la mayor parte de los servicios se ejecutan como proceso de usuario en lugar de como proceso del kernel.

Una de las principales desventajas esta solución es que puede tener un rendimiento más bajo que las otras

soluciones debido a la carga de procesamiento adicional impuesta por las funciones de los sistemas.[2]

2.3.3 Módulos

Una de las mejores metodologías para diseñar sistemas operativos es la de un kernel modular, que permite cargar módulos[2] dinámicamente durante el arranque o en tiempo de ejecución.

El kernel modular permite añadir controladores de bus y dispositivos. Cada sección del kernel posee interfaces bien definidas y protegidas es más flexible que un sistema de niveles porque permite comunicación entre módulos, sin embargo es más eficiente que un microkernel, debido a que los módulos no necesitan invocar un mecanismo de paso de mensajes para comunicarse.[2]

2.4 Planificación del Sistema Operativo

Los mecanismos de planificación de la CPU son la base de los sistemas operativos que dan soporte a la multiprogramación,

[2] Módulo: Parte del código que tiene como objetivo cumplir una o varias tareas que le asigne el sistema el sistema operativo.

mediante la asignación de la CPU entre distintos procesos con el fin de maximizar el uso del CPU.

A continuación se explica uno de los conceptos más importante en la en la planificación de sistemas operativos como es el algoritmo de afinidad que permite mantener equilibrada la actividad de los distintos CPU.

2.4.1 Afinidad de Procesos

Cuando un proceso se ha estado ejecutando en un procesador específico, los datos a los que el proceso ha accedido se almacenaran en la cache del procesador y los futuros accesos a memoria por parte del proceso consultaran a la memoria cache del procesador, ahora si se migra un proceso a otro procesador los contenidos de la memoria cache del procesador origen deben ser liberados y la cache del procesador destino deben ser llenada. [2]

Debido al alto tiempo de procesamiento la mayoría de los sistemas intentan evitar la migración de procesos de un procesador a otro e intenta mantener en ejecución cada

proceso en el mismo procesador .Esto se conoce como afinidad de procesos, lo que significa que un proceso tiene afinidad hacia el procesador en que está ejecutándose actualmente.

2.5 Sincronización de Procesos

La sincronización es la transmisión y recepción de señales que tiene por objetivo llevar a cabo el trabajo de un grupo de procesos cooperativos[3]. La sincronización entre procesos es necesaria para prevenir y o corregir problemas comunes de consumidor y productor debido al acceso concurrente de recursos compartidos, tales como estructuras de datos o dispositivos de entrada salida, de procesos cooperativos.

Como ejemplo, supongamos que tenemos tres procesos concurrentes que quieren modificar un mismo archivo. Si los tres acceden a este al mismo tiempo el archivo podrá contener datos incorrectos. Este tipo de problemas ocurre frecuentemente en los sistemas operativos y debido a la importancia de este problema en

[3]Un proceso es cooperativo si puede afectar o ser afectado por los otros procesos que se están ejecutando en el sistema.

esta sección se dedica a la sincronización y coordinación de procesos.[2]

Sin una sincronización adecuada entre procesos, la actualización de variables compartidas puede inducir errores de consistencia de datos. Una de las causas principales de este problema es que procesos concurrentes puedan observar valores temporalmente inconsistentes de una variable compartida mientras se actualizan.[3] Existen varias razones para realizar concurrencia:

- Comparten recursos físicos.
- Recursos lógicos
- Aceleran los cálculos de procesamiento.

2.5.1 Sección Crítica

El problema de la sección crítica consiste en diseñar un protocolo para que los procesos puedan acceder en forma ordenada a recursos compartidos y poder resolver problemas

como la correcta actualización de de registros. Cualquier solución al problema de la sección crítica deberá satisfacer los tres requisitos siguiente: [2]

- **Exclusión mutua:** Si un proceso se está ejecutándose en su sección crítica, los demás procesos no pueden estar ejecutando sus secciones críticas.

- **Progreso:** Si ningún proceso está ejecutando su sección crítica, y algunos procesos desean entrar en sus correspondientes secciones críticas, sólo aquellos procesos que no estén ejecutando sus secciones restantes pueden participar en la decisión de cuál será el siguiente que entre en su sección crítica, y esta selección no se puede posponer indefinidamente.

- **Espera limitada:** Existe un límite en el número de veces que se permite que otros procesos entren en sus secciones críticas después de que un proceso

haya hecho una solicitud para entrar en su sección crítica y antes de que la misma haya sido concedida.

Los principales mecanismos de sincronización que ofrecen los sistemas operativos son:

- Señales
- Tuberías
- Semáforos
- Mutex y variables condicionales
- Paso de mensajes

En nuestro proyecto se utilizan semáforos por lo cual se describe en la siguiente sección este mecanismo con mayor detalle.

2.5.2 Semáforos

Un semáforo es una variable especial del sistema operativo para restringir o permitir el acceso a recursos compartidos, donde se manipulan variables o recursos que deben ser accedidos de forma especial, como por ejemplo, un recurso de almacenamiento del sistema, en un entorno de multiprocesamiento.

Cuando se inicia una operación con un semáforo, el sistema operativo no permite que otro proceso puede tener acceso al semáforo hasta que la operación termine o se bloqueó, esto es importante para resolver problemas de sincronización y evitar condiciones de carrera “cuando dos o más procesos leen o escriben en un área compartida y el resultado depende de los instantes de ejecución de cada uno”. [4]

Un tipo simple de semáforo es el binario, que puede tomar dos valores: 0 y 1. Si, para un semáforo binario, S = 1 entonces el recurso está disponible y la tarea lo puede utilizar; si S = 0 el recurso no está disponible y el proceso debe esperar..

Los semáforos se implementan con una cola de tareas o de condición a la cual se añaden los procesos que están en espera del recurso.[2]Sólo se permiten tres operaciones sobre un semáforo:

- Inicializar
- Espera (wait)
- Señal (signal)

A continuación se explicara el problema del consumidor productor.

El problema Productor/Consumidor consiste en el acceso concurrente por parte de procesos productores y procesos consumidores sobre un recurso común que resulta ser un buffer de elementos. Los productores tratan de introducir elementos en el buffer de uno en uno, y los consumidores tratan de extraer elementos de uno en uno.

La solución a este problema se obtiene con la introducción de tres semáforos como se menciona en: [5]

Uno controlará el acceso al buffer y se encargará de garantizar la exclusión mutua.

El segundo semáforo lo utilizará el proceso productor para controlar la posibilidad de acceso o no al buffer (Cuando el buffer esté lleno no deberá poder insertar elementos).

El último semáforo realizará tendrá la misma misión que el anterior, controlará los accesos del proceso consumidor al buffer, para evitar acceso de este cuando el buffer esté vacío.

Tanto el proceso productor como el consumidor tendrán la posibilidad de producir o consumir elementos, pero esto no lo harán siempre. Cuando un proceso decide consumir o producir un elemento, primero comprobará si puede hacerlo, y esto lo hará consultando el buffer. Para la consulta del buffer deberá crear la exclusión mutua, Si puede hacerlo

(consumir o producir) seguirá en la sección crítica y recogerá o pondrá un elemento del buffer. Hecho esto, comprobará que si el otro proceso está esperando la operación que ha realizado (el otro proceso haya realizado una operación P(s)), si es así le avisará de que ya puede continuar su tarea (realiza una operación V(s) sobre él).

Si el proceso no puede trabajar con el buffer (porque este está lleno o vacío), realizará una operación P(s), en espera de que el otro proceso le libere de la incapacidad de trabajar mediante su trabajo con el buffer (realice una operación V(s)).

Tanto si los procesos consumen o producen elementos, como si no lo hacen, cuando terminen su consulta se volverán a conectar con el buffer para hacer que termine la exclusión mutua, lo que colocara el semáforo del buffer de nuevo en verde y dispuesto para servir otra consulta.

Con esto se conseguirá que el productor produzca elementos y que el consumidor los consuma.

CAPÍTULO 3

3 CÁMARAS DE VIDEO

Son dispositivos utilizados para capturar imágenes, utilizan como plano de enfoque un sensor CCD (Charge coupled device), que es un chip sensible a la luz, electrónico, con una superficie fotosensible que reacciona a la luz cuando esta pasa a través del objetivo de la cámara.

Cada uno de los miles o millones de diminutos pixeles que componen en sensor CCD convierte esta luz en electrones. El número de electrones, generalmente conocido como carga acumulada de pixeles, se mide y, a continuación, se convierte en un valor digital.

La información que se obtiene del sensor puede estar codificada en formato yuyv, rgb24, rgb32, etc. y esta puede ser guardada en distintos formatos digitales como lo son el RAW, TIFF, JPG, PPM.

Figura 3.1 Sensor CCD

La clasificación de las cámaras puede hacerse atendiendo a varios criterios: su forma, el formato, el sistema de visión, su uso, interfaz etc.

Uno de los aspectos claves de un sistema, es la elección del tipo de conectividad entre la cámara y el dispositivo de adquisición pues estas tienen diferentes características que afectan al rendimiento del sistema, como son las velocidades de transmisión y los costes.

Existen múltiples tipos de conexiones adoptados por la industria de visión artificial, entre las más tradicionales se encuentran el estándar video analógico o el RS422 y LVDS aunque cada vez están siendo menos utilizados.

Los estándares de conexión más comunes actualmente en el mercado son: Camera link, USB 2.0, IEEE 1394a y IEEE1394b (firewire) y GigEVision (gigabit Ethernet). Cada uno de estos estándares ofrece diferentes ventajas y desventajas, con lo cual los requisitos específicos de cada sistema definirán cual es la aplicación más adecuada. En este trabajo se estudian solamente aquellos estándares utilizados en nuestro experimento (Firewire, USB). Para más información de estos estándares de conexión

Figura 3.2 Tipos de conectores para transmisión de video

3.1.1 Firewire

Se denomina Firewire al tipo de puerto de comunicaciones de alta velocidad desarrollado por la compañía Apple. La denominación oficial de esta interfaz es IEEE 1394. Se trata de una tecnología para la entrada/salida de datos en serie a alta velocidad y la conexión de dispositivos digitales. Esta interfaz se caracteriza principalmente por:

- Su gran rapidez, siendo ideal para su utilización en aplicaciones multimedia y almacenamiento, como videocámaras, discos duros, dispositivos ópticos, etc., debido a que alcanzan una velocidad de 400 megabits por segundo.

- Necesidad de una tarjeta de adquisición.

- Flexibilidad de la conexión y la capacidad de conectar un máximo de 63 dispositivos.

- Garantiza una distribución de los datos en perfecta sincronía.

- Versiones de 4, 6, 9, 12 pines.

- Ancho de banda de 50 – 400 MB/s

- Velocidad de transferencia de datos de 25 MB/s

3.1.2 USB

Sin duda, el estándar de conexión más extendido. Fue diseñado como un interfaz para periféricos de ordenadores, y no fue diseñado originalmente con la intención de ser usado en transferencias de datos a alta velocidad. A pesar de ello, ha sido adoptado como una conexión estándar en cámaras, y sirve adecuadamente en muchas aplicaciones.

El interfaces USB 2.0 ofrece trasferencias de datos de 480 Mbit/s, lo que teóricamente es igual a 60 MB/s. En realidad la máxima velocidad real de trasferencia de datos que se puede lograr de forma sostenida a través de USB 2.0 es del orden de 45 MB/s, lo cual es suficiente para muchas aplicaciones. Por ejemplo, una cámara monocroma de 8 bits de 640x480 píxeles, funcionando a 30 cuadros por segundo

requiere solamente 9 MB/s; una cámara RGB color de 640x480 píxel a 25 fps requiere aproximadamente 27 MB/s.

Sin embargo, el USB 2.0 tiene limitaciones y no puede ser usado con cámara de alta velocidad y resolución, o en aplicaciones de alta carga computacional o de respuesta en el tiempo critica. No obstante, por facilidad de instalación, bajo coste y acceso a casi cualquier ordenador sin la necesidad de hardware adicional hace que el USB 2.0 sea una elección adecuada para muchos casos.

3.2 Parámetros de una Cámara

Una cámara tiene parámetros intrínsecos como extrínsecos (ó internos y externos). Los parámetros intrínsecos son aquellos referentes a la propia cámara como el tamaño de los pixeles, distancia focal, contraste, brillo, etc. Los parámetros extrínsecos son aquellos que definen la posición y la orientación del cuadro de referencia de la cámara con respecto al mundo real, es decir, dan la orientación externa de la cámara.

Para la implementación de este proyecto, se considerará los parámetros de brillo, contraste y balance de blancos. El brillo y el

contraste son parámetros básicos, ya que una modificación de cualquiera de ellos suele suponer un cambio visible en la imagen como se puede apreciar en las figuras 3-3 y 3-4. El parámetro brillo permite corregir problemas de iluminación en imágenes oscuras o demasiado claras, mientras que el contraste permite realzar los colores de una imagen. Adicionalmente, el parámetro balance de blancos es un control de la cámara que sirve para ajustar el brillo de los colores básicos rojo, verde y azul (RGB) con el objeto de que la parte más brillante de la imagen aparezca como color blanco, y la menos brillante como negro. El ajuste de balance de blancos es necesario cuando existe una iluminación artificial que produce que uno de los componentes de color prevalezca sobre las otras, por ejemplo, cuando una escena está con una iluminación basada en bombillas incandescentes, esta tendrá un color rojo predominante.

Imagen con brillo 0

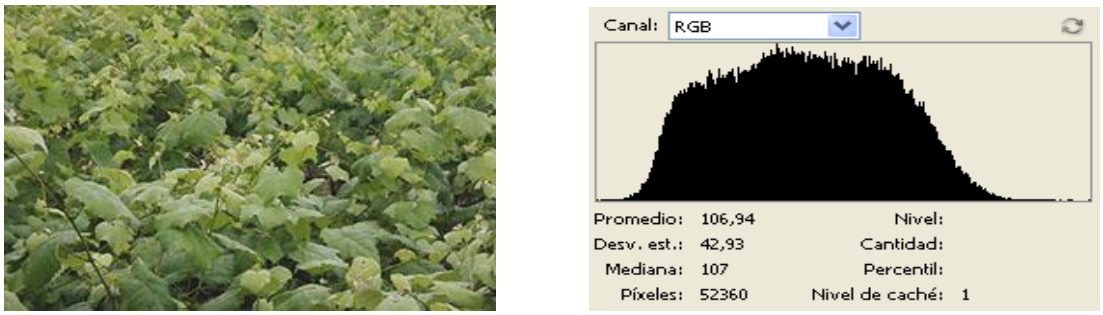


Imagen con brillo 40

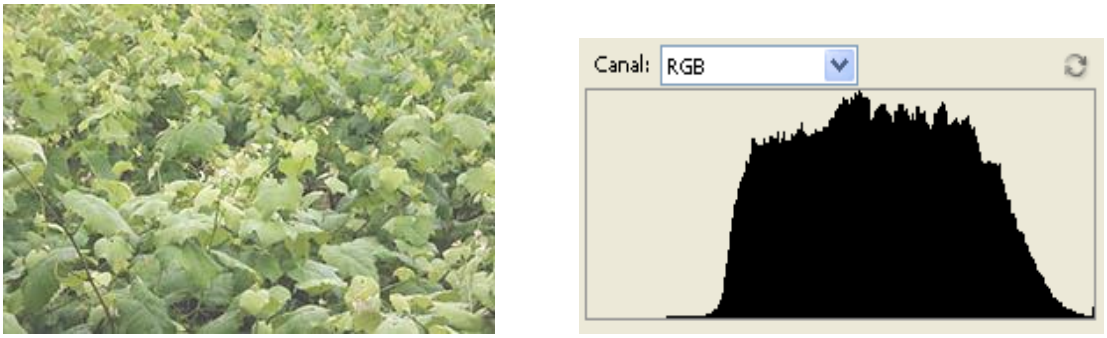


Imagen con brillo 90

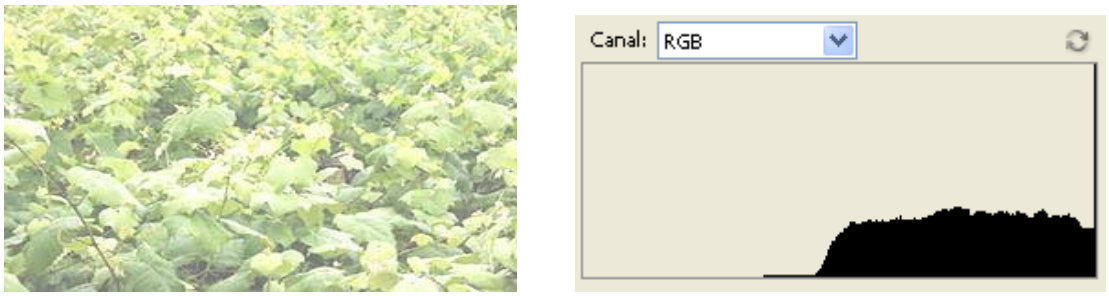


Figura 3.3 Imagen con diferentes niveles de brillo

Imagen con contraste -50

Imagen con contraste -33

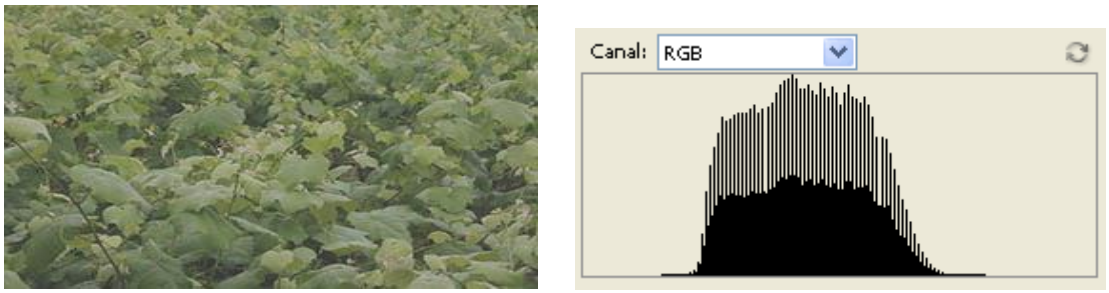


Imagen con contraste +33

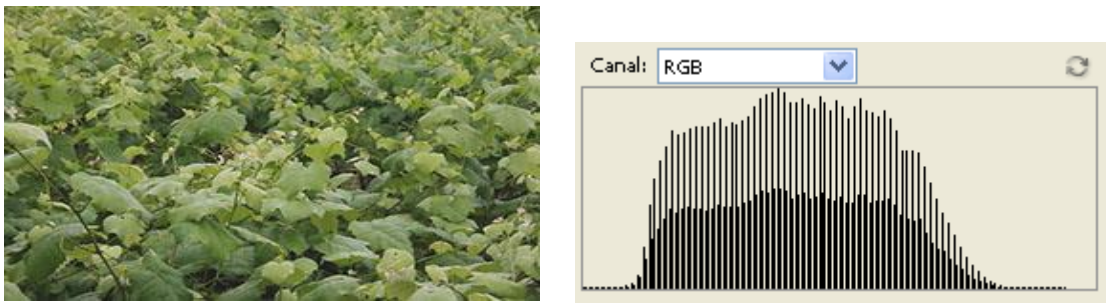


Figura 3.4 Imagen con diferentes niveles de contraste

3.3 Calibración

La calibración de una cámara es el proceso que permite estimar los valores de los parámetros intrínsecos y extrínsecos que definen las condiciones de formación de la imagen dentro del campo de visión.

El uso de cámaras calibradas puede ser útil para resolver una serie de problemas relacionados con la obtención de la posición 3D de los objetos en el espacio a partir de sus imágenes o para reconstrucción tridimensional. Esto puede permitir, entre otras tareas, la realización de mapas del entorno de la cámara, el seguimiento de un objeto específico o la obtención de la posición de la cámara respecto a objetos que la rodean. También puede facilitar la navegación por su entorno de un robot móvil, permitiendo evitar obstáculos, dirigirse a objetos determinados o facilitar la definición de la trayectoria más adecuada para alcanzar su destino. [6]

La calibración de una cámara es un problema complejo de solucionar ya que muchos son los factores a resolver que influyen en los resultados. Esta complejidad se reduce por parte de los modelos clásicos de calibración debido a que emplean modelos de cámaras ideales o simplificadas de sus equivalentes físicos. Uno de los factores más importantes a considerar en el proceso de calibración de una cámara son las posibles fuentes de ruido que influyen en el proceso de formación de la imagen que se debe, la mayoría de veces, al equipo electrónico utilizado para la captación de imágenes y a las posibles interferencias a la hora de transmitir los bits de información, sin embargo, existen algoritmos como el filtro medio y

mediano que son normalmente usados para eliminar el ruido de una imagen digital.

3.3.1 Métodos de Calibración

Existen diversas técnicas de calibración de cámara. De acuerdo a la literatura estas pueden ser clasificadas en dos grandes categorías: Calibración fotogramétrica y auto calibración. [7]

La calibración fotogramétrica se realiza mediante la observación de patrones cuya geometría en el espacio 3D es conocida con un buen nivel de precisión. Los patrones de calibración normalmente están posesionados en dos o tres planos ortogonales entre ellos. En algunos casos, basta con un único plano, cuya traslación es perfectamente conocida. Este tipo de calibración requiere una configuración elaborada y una costosa preparación del equipo. Por otro lado, el método de auto calibración se basa en el movimiento de la cámara observando una escena estática, a partir de sus desplazamientos y usando la información de la imagen. La

rigidez de la escena impone en general restricciones sobre los parámetros de la cámara. [8]

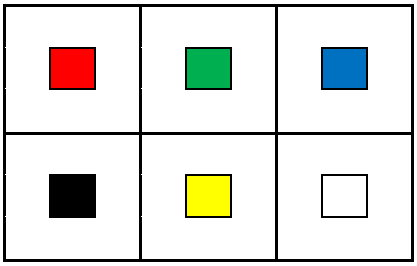
3.4 Almacenamiento de una Imagen

Para la implementación de este proyecto, se eligió el formato de imagen PPM (Portable Pixel Map) por tratarse de un formato normalmente utilizado para realizar prácticas de procesamiento de imágenes debido a que es un formato sin comprensión y además puede almacenar enformato de texto, por lo que no es necesario realizar un tratamiento binario del fichero donde se almacena la imagen. [9]

Un fichero de imagen en formato PPM comienza con una cabecera con un identificador (P3) que indica el tipo de fichero (pixmapascii), seguido por tres números, que representan el número de pixeles horizontales, el número de pixeles verticales y el máximo valor de cada componente RGB de la imagen. A continuación se especifican, por filas, los tres valores de cada componente RGB (Red, Green, Blue) de cada pixel que estarán comprendidos en un rango de valores siendo cero el mínimo valor y el máximo valor 255. [10]

Por ejemplo el fichero ppm representada en la Figura 3.5 donde observamos un archivo ppm de dos filas tres columnas.

Imagen



Archivo

```
P3
 3      2
255

255    0      0            0    255    0            0      0    255
  0    0      0          255    255    0          255    255    255
```

Figura 3.5 Representación de un fichero ppm

CAPÍTULO 4

4 DISEÑO E IMPLEMENTACIÓN

En este capítulo se realiza un análisis del diseño del sistema y de los componentes utilizados durante la implementación del mismo. En la Figura 4.1 se muestra un diagrama del funcionamiento general del sistema y más adelante se explicara con mayor detalle cada sección.

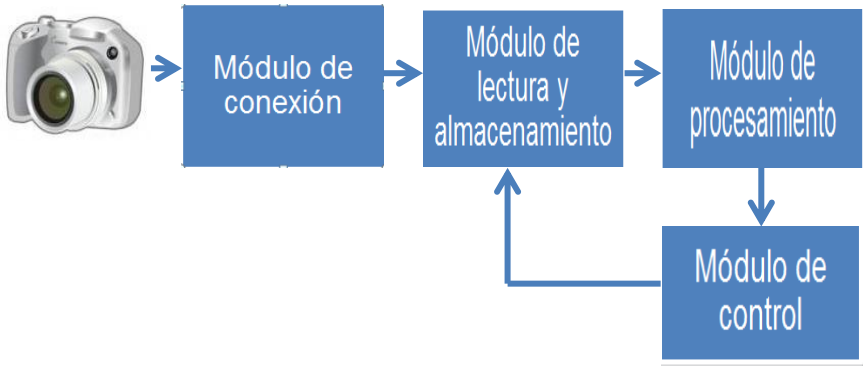


Figura 4.1 Módulos del sistema

Como se aprecia en la Figura 4.1 sistema está dividido en los siguientes módulos:

- Módulo de conexión

- Módulo de lectura y almacenamiento

- Módulo de procesamiento

- Módulo de control

Adicionalmente, el sistema está diseñado de tal forma de que el primer módulo permite la conexión con el driver de la cámara utilizando la librería de V4l2 que viene implementado como modulo del kernel. Para aprovechar el hardware del computador, a cada uno de los tres módulos siguientes le corresponde un hilo de ejecución y dado que los hilos van a compartir información es necesaria la utilización de variables compartidas. Para la creación y control de los hilos se utilizó la librería ***pthread*** que es una librería que cumple con los estándares POSIX y que permite ejecutar varios hilos de forma concurrente.

4.1 Módulo de Conexión

Este módulo establece la comunicación con la cámara, lo cual encierra una secuencia de pasos mostrados en la Para establecer la comunicación con la cámara es necesaria la librería videodev2.

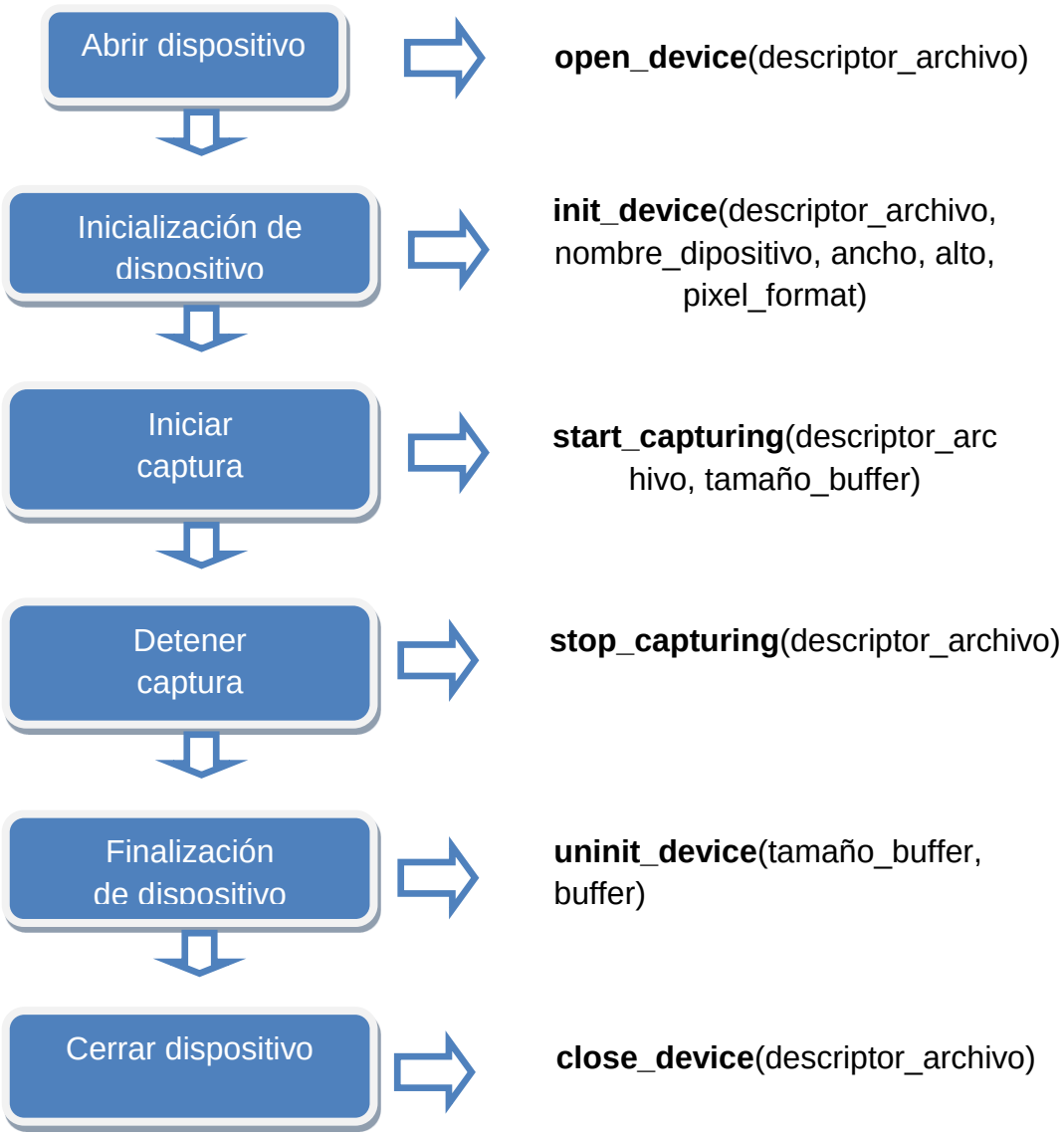


Figura 4.2 Módulo de conexión con el driver del dispositivo

4.1.1 Abrir y Cerrar Dispositivo

Las funciones ***open_device( )*** y ***close_device( )*** son aquellas que permiten abrir y cerrar la conexión con el módulo VIDEODEV del kernel, para lo cual, primero se crea un puntero que hace referencia al nombre del archivo de la

cámara (dev/videoX) que se crea automáticamente cada vez que se conecta un dispositivo al computador, donde X es un índice que indica el número de cámaras más uno conectadas a la computadora. Para abrir el dispositivo se utiliza la función ***open( )***que recibe como parámetros el nombre del dispositivo, el tipo de acceso que puede ser de lectura, escritura ó de lectura y escritura. Este sistema solicita al usuario el nombre del dispositivo con el cual se desea establecer la conexión.

Para cerrar el dispositivo se utiliza la función ***close ( )*** que libera el descriptor de archivos.

4.1.2 Inicialización del Dispositivo

Debido a la variedad de dispositivos de video, se debe verificar que la cámara conectada sea compatible con el controlador de V4L2, para esto, se realiza una llamada al sistema con la función ***ioctl ( )*** que recibe como parámetro el descriptor de archivos, el tipo de requerimiento de V4L2: VIDIOC_QUERYCAP, que es el encargado de consultar si el dispositivo es compatible con V4L2 y un puntero hacia la estructura V4L2_capability que es llenado con información

acerca del controlador y el hardware del dispositivo. Cuando estructura se encuentra llena podemos acceder a las funciones y métodos dela misma para cambiar o verificar ciertas propiedades del dispositivo, como la selección de entrada de video y el método de transmisión de video para lo cual se utilizan las siguientes banderas:

- **V4L2_CAP_VIDEO_CAPTURE:** Permite veirificar si el disposistivo soporta captura de video.

- **V4L2_CAP_STREAMING:**Permite verificar si el dispositivo soporta el streaming de video E/S.

En esta etapa se define el formato y el diseño de la imagen en memoria, para ello utilizamos la estructura V4L2_pix_format que nos permite definir el ancho, alto y el formato de pixel de la imagen, en nuestro proyecto utilizamos el formato de pixel YUV porque este formato separa los componentes de luminancia(Y) y crominancia(UV) y debido a que la visión humana es mucho más sensitiva a las variaciones de intensidad que a las variaciones de color resulta un formato mas adecuado porque provee un ancho

de banda mayor para la informacion de luminancia que para la de crominancia. Cuando el dispositivo no es compatible, la función ***ioctl ( )*** devuelve un código de error EINVAL (argumento inválido).

4.1.3 Iniciar Captura

En esta etapa se crea el tipo de dato ***struct buffer*** que es la que almacenará cada uno de los fotogramas capturados. A cada región del buffer se le asigna memoria con el método MemoryMmap pasándole como parámetro de entrada V4L2_BUF_TYPE_VIDEO_CAPTURE debido a que es la cámara es un dispositivo de captura.

Luego de realizar la captura se procede a libera la memoria del buffer y cerrar conexión con el dispositivo.

4.2 Módulo de Lectura y Almacenamiento

En este móduloes donde se realiza la lectura, conversión y almacenamiento físico de los frames almacenados en memoria. Inicialmente contamos con un buffer que contiene los frames con codificación de imagen YUV. V4l2 también permite la codificación de

imagen en RGB, pero se eligió el modelo de YUV debido a que este formato concentra la comprensión en el color, preservando la luminosidad lo cual es más conveniente porque el ojo humano es más sensible a la luminosidad que al color logrando reducir el tamaño de la captura sin afectar demasiado el color, lo que se traduce en que los frames ocuparan menos espacio y por lo tanto se transmitirán de una manera más rápida. Este modelo define un espacio de color en términos de una componente de luminancia y dos componentes de crominancia. El parámetro Y representa la luminancia (es decir, información en blanco y negro), mientras que U y V representan la crominancia (es decir, información con respecto al color).

Una vez leído cada frame, es necesario realizar la conversión de la imagen a RGB porque este espacio de color permite representar gran parte de la gama de colores perceptibles por el ojo humano.

Para realizar la trasformación de YUYV a RGB es necesario tener en cuenta que cada cuatro bytes representan dos pixeles de RGB, es decir que a cada pixel le corresponde un valor de Y (luminancia) la combinación de U y V (crominancia). [11]

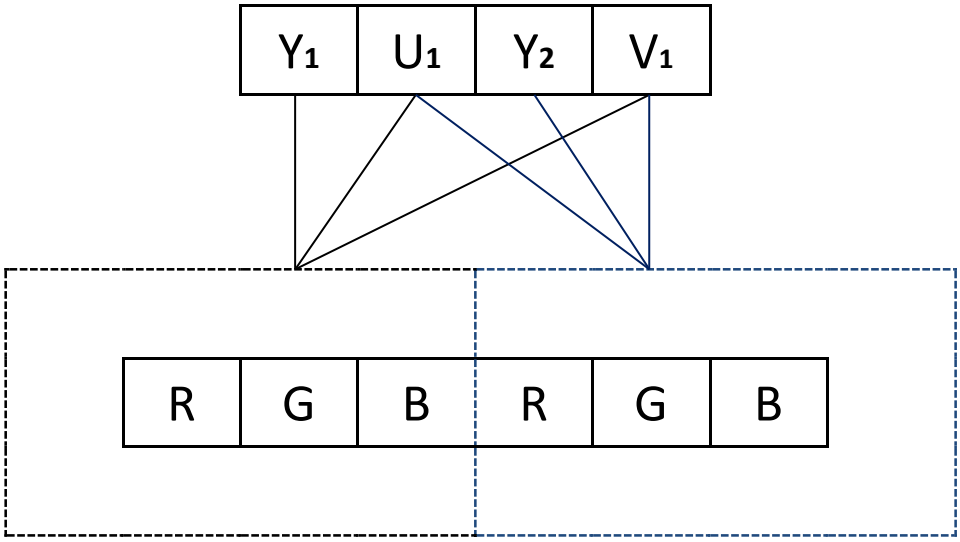


Figura 4.3 Formato YUV 4:2:2

Lo siguiente para culminar con la transformación, es multiplicar por un factor a los valores de luminancia y crominancia ese factor permite tener la proporción exacta de cantidad de luz que se necesita en cada color para representar un pixel en las siguientes ecuaciones se muestran las conversiones para obtener los diferentes canales de un pixel en RBG.

$$B = 1.164 * (y - 16) + 2.018 * (v - 128)$$

Ecuación 4-1 Conversiones de espacio de color YUV al canal B.

$$G = 1.164 * (y - 16) - 0.813 * (u - 128) - 0.391 * (v - 128)$$

Ecuación 4-2 Conversiones de espacio de color YUV al canal G.

$$R = 1.164 * (y - 16) + 1.596 * (u - 128)$$

Ecuación 4-3 Conversiones de espacio de color YUV al canal R.

4.3 Módulo de Procesamiento y Control

A continuación se presenta una guía para la calibración de los parámetros de brillo, contraste y balance de blancos.

El método descrito está basado en el trabajo de Harsh Nanda propuesto en el artículo [12]en el cual se presenta un método simple y eficaz para ajustar automáticamente los parámetros de brillo, contraste y balance de blancos en una cámara y luego propagar esos valores en el resto de cámaras y así suavizar los límites de una imagen panorámica obtenida por la unión de las capturas de todas las cámaras.

A este algoritmo se le dio un enfoque diferente para que pueda ser aplicado a la implementación de nuestro proyecto. Inicialmente, se cuenta con varias fotos adquiridas manualmente durante el transcurso del día que corresponderán a las “mejores fotos” posibles obtenidas por inspección visual. A continuación, para la calibración del parámetro brillo y contraste se procede a obtener el promedio de

una de las imágenes ideales para posteriormente ir cambiando el parámetro brillo o contraste en la cámara hasta que el promedio de la imagen recientemente capturada sea igual al promedio de la imagen ideal. Para el proceso de calibración de balance de blancos, el proceso es similar, pues también es necesario obtener el promedio de la imagen pero de cada canal por separado e ir manipulando el parámetro de ganancia de rojos y azules hasta que se cumplan las siguientes igualdades:

$$\frac{Promedio\ rojos\ img\ capturada}{Promedio\ de\ verdes\ img\ capturada} = \frac{Promedio\ rojos\ img\ referencia}{Promedio\ verdes\ img\ referencia}$$

$$\frac{Promedio\ azules\ img\ capturada}{Promedio\ de\ verdes\ img\ capturada} = \frac{Promedio\ a\ img\ referencia}{Promedio\ verdes\ img\ referencia}$$

Ecuación 4-4 Promedio de la imagen para la calibración de balances de blancos.

Con este método logramos que exista una calibración automática de brillo, contraste y balance de blancos de una imagen en tiempo real, lo que evita un post procesamiento y por lo tanto nos permite tener un tiempo de respuesta mínimo.

4.4 Hilos de Ejecución

A excepción del módulo de conexión, a cada módulo del sistema le corresponde un hilo de ejecución. Estos pueden ejecutarse en un solo procesador o en procesadores diferentes, esto depende de la capacidad del CPU donde se esté procesando el programa. En el Anexo A se muestra un gráfico donde se puede visualizar la comunicación de los diferentes hilos del sistema, a continuación se explica cada uno de ellos.

En el programa se crean tres hilos de ejecución: camera, processing_algorithm y control_camera. Además se utilizan variables de condición y semáforos para la sincronización y comunicación de los hilos.

Debido a que accedemos a un descriptor para obtener y configurar los parámetros de la cámara, además de realizar diferentes operaciones sobre las imágenes capturadas, no se pueden realizar tareas de lectura y escritura al mismo tiempo ya que esto provocaría perdida de información. El sistema maneja dos estados:

- **Free:** es el estado inicial de la imagen y se encuentra con formato de pixel YUV.

- **Busy:** es el estado de la imagen después de realizar la transformación del formato YUV a RGB y está lista para ser procesada.

Un frame inicialmente se encuentra en estado *Free*y el hilo processing_algorithm*se encuentra bloqueado hasta* que el hilo *camera* termine la transformación de la imagen al formato RGB, cambie el estado de la imagen a busy y envía una señal a processing_algorithm para que pueda seguir con su ejecución.

De la misma manera el hilo de *control camera* se encuentra bloqueado por el hilo de *processing_algorithm* para que no pueda cambiar los valores de los parámetros de la imagen hasta que este calcule el valor promedio de los frames.

Además, mientras en hilo de *camera* se encuentre leyendo desde el descriptor de archivos este bloqueara al hilo de *control_camara*para que no realice escritura sobre el mismo.

Para futuras comparaciones de rendimiento este sistema fue desarrollado con dos librerías diferentes: V4L2 y OpenCV que corresponden al programa A y programa B respectivamente.

CAPÍTULO 5

5 EVALUACIÓN DEL DESEMPEÑO DEL SISTEMA

5.1 Metodología de Evaluación

Con los recursos de implementación descrita en el capítulo 4, se realizaron varios experimentos que permiten evaluar la eficiencia de la solución[4] y la exactitud del algoritmo[5] utilizado en el sistema.

Las pruebas utilizadas para evaluar el sistema se llevan a cabo en un espacio de trabajo controlado por varias condiciones que se listan a continuación:

[4] Tiempo requerido para la calibración de los parámetros de la cámara, desde que el hilo del algoritmo detectó una diferencia de promedios entre la imagen actual y la imagen de referencia, por una variación de luz, hasta que se calibraron los parámetros de brillo y contraste.
[5] Robustez del algoritmo a cambios de iluminación.

- **Condiciones de iluminación y ambientales:** debido a que las pruebas se realizaron en exteriores, se toma en consideración, la hora del día en la que se realizaron las pruebas y las condiciones ambiéntales de ese día. El rango de horas normalmente oscila entre las 8:00 y 18:30 y las condiciones ambientales del lugar en el momento que se realizaron los experimentos fueron nublado y parcialmente nublado.

- **Resolución de captura:** el tamaño de la imagen capturada es de 640*480 con una resolución de 5 megapíxeles.

Para medir la eficiencia del sistema propuesto se diseñan pruebas que evalúan el tiempo de ejecución del sistema y la robustez del algoritmo a cambios de iluminación.

- **Eficiencia de la solución:** Se mide el tiempo que le toma al sistema procesar datos desde que detecta una variación en el ambiente hasta que calibra los parámetros de brillo y contraste. Esta medición se la realiza en dos formas:

1. Se compara el tiempo de ejecución al cambiar el número de procesadores asignados al sistema.

2. Se compara el tiempo de ejecución utilizando el programa A y el programa B descritos en la sección 4.4.

- **Exactitud del algoritmo de calibración:** Se mide la similitud entre la imagen calibrada y la imagen de referencia, calculada en base a los promedios de las imágenes y la comparación de sus histogramas6.

5.2 Experimentos y Resumen

5.2.1 Eficiencia de la Solución

Los siguientes experimentos fueron realizados utilizando el enfoque en la sección 5.1

[6] Representación gráfica de una variable en forma de barras, donde la superficie de cada barra es proporcional a la frecuencia de los valores representados. En el eje vertical se representan las frecuencias y en el eje horizontal los valores de las variables.

Experimento 1

Para este experimento las condiciones ambientales de la captura de la imagen de referencia fueron las mismas utilizadas en la calibración del algoritmo. Del mismo modo, se usó la misma imagen de referencia para la ejecución del algoritmo con diferentes números de procesadores, esto se realizó para obtener la diferencia entre los distintos tiempos de ejecución con uno o varios procesadores.

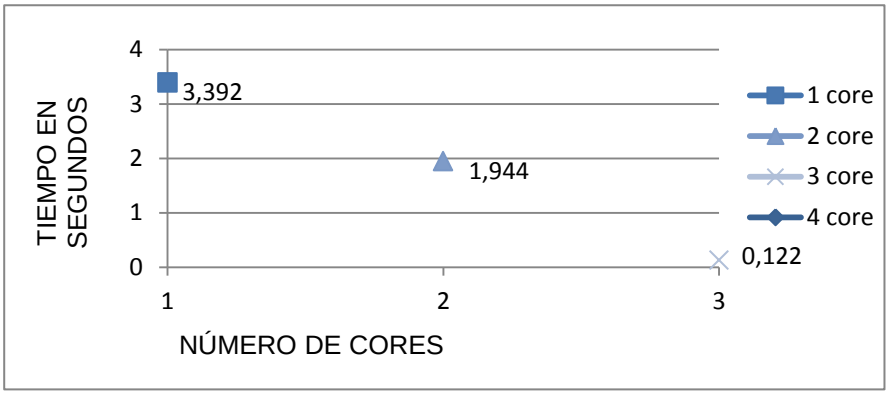


Tabla 5-1 Tiempo de procesamiento vs número de procesadores del CPU

Al asignarle los tres hilos (*algorithm, camera, control_camara*) a un sólo procesador el tiempo de ejecución es mayor debido a que todo el trabajo obviamente es gestionado por un solo CPU y por lo tanto, los bloqueos y

periodos de espera son más frecuentes debido a las posibles interrupciones provenientes de otros procesos que requieren el uso del mismo CPU.

Por el contrario, si a cada hilo le asignamos un procesador diferente, el tiempo de ejecución se reduce 96% lo cual es lógico porque el trabajo se encuentra distribuido en varios procesadores. Adicionalmente, el hecho de que cada hilo se ejecute en un procesador ayuda a incrementar los aciertos de caché y eliminar el consumo de la infraestructura de cambio entre hilos.

Experimento 2

Para este experimento los programas A y B fueron ejecutados bajo las mismas condiciones ambientales y escenarios utilizando la misma imagen de referencia. El objetivo de este experimento es comparar los tiempos de ejecución del programa usando diferentes librerías.

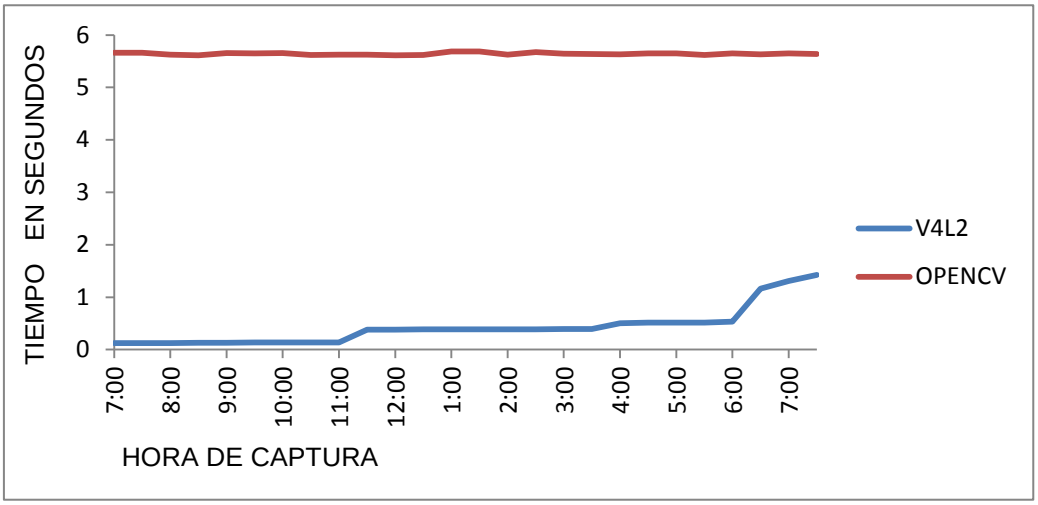


Figura 5.1 tiempos de procesamiento en V4L2 vs OPENCV

En la figura 5.2 se puede apreciar que utilizando el programa A (librería V4L2) los tiempos de respuesta son menores a los tiempos de respuesta del programa B (librería OpenCV), esto se debe a que la librería de V4L2 trabaja en un lenguaje de bajo nivel, no utiliza librerías externas y permite la comunicación directa con el kernel de Linux.

5.2.2 Exactitud del Algoritmo de Calibración

En esta sección se procede a evaluar la exactitud del algoritmo a los diferentes cambios de luz ambiental.

Por facilidad para el desarrollo de este experimento se utilizó la imagen de referencia y las imágenes resultantes procesadas en el programa A en la sección 5.1 experimento dos. Posteriormente se obtuvo el histograma de cada una de las imágenes y se utilizó la métrica de χ^2 (chi cuadrada) para saber si existía o no relación entre el histograma de la imagen de referencia y el histograma de cada imagen resultante. La distribución *Chi cuadrado* permite calcular si los resultados estadísticos de un experimento se alejan significativamente o no de los resultados esperados del modelo teórico.

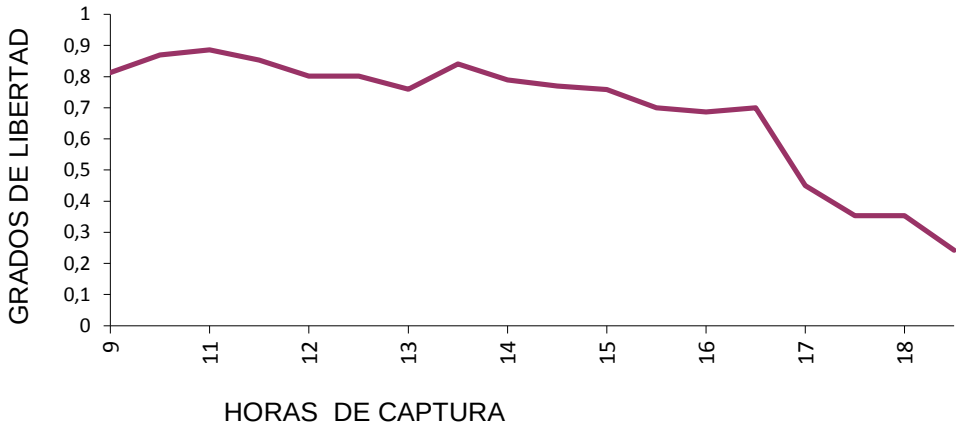


Figura 5.2 Grados de libertad entre la imagen de referencia y cada imagen resultante

En la figura 5.3 se graficaron los resultados obtenidos por chi cuadrada, entre más alto sea el valor obtenida por la chi cuadrada mayor relación existe entre el histograma perteneciente a la imagen de referencia y el histograma de las imágenes resultantes del algoritmo de calibración de la cámara, por lo que se puede concluir que en condiciones normales de iluminación se cumple que a medida de que la luz disminuye el algoritmo es menos exacto con respecto a la imagen de referencia.

En estas pruebas se utilizaron los histogramas debido a que permite representar la cantidad de pixeles que aparecen en una imagen con cada determinado valor de luminosidad, lo cual también nos indica que el valor promedio de la imagen ya que esta es la suma de valores de luminosidad de cada pixel divido para el número de pixeles, lo cual indica si dos imágenes poseen el mismo valor promedio no exactamente son imágenes iguales, pero sí que posean el mismo valor de luminosidad.

A continuación se muestran algunas imágenes resultantes del experimento número dos, se puede observar que a

medida que oscurece y debido a la falta de luz en la escena el resultado del algoritmo de calibración no es bueno, ya que bajo estas condiciones aunque los parámetro de brillo y contraste cambien no es posible que el valor promedio de la imagen sea igual al parámetro de la imagen de referencia.

a) Imagen de Referencia

b) 8:30

c) 11:30

d) 15:00

e) 17:30

f) 19

Figura 5.3 Imágenes Resultantes de la calibración La imagen en un rango de horas

CONCLUSIONES

1. Las condiciones ambientales afectan notablemente el proceso de calibración de la cámara a un punto donde la calibración ya no es posible.

2. La ejecución del sistema se realiza en promedio cinco segundos más rápido que si utilizáramos una librería externa como OpenCV.

3. La programación concurrente permite reducir los tiempos de ejecución sobre plataformas multiprocesadoras.

RECOMENDACIONES

1. Para mayor aprovechamiento del sistema tener en cuenta las características del CPU ya que el sistema se encuentra diseñado de tal manera que permite aprovechar la afinidad del procesador.

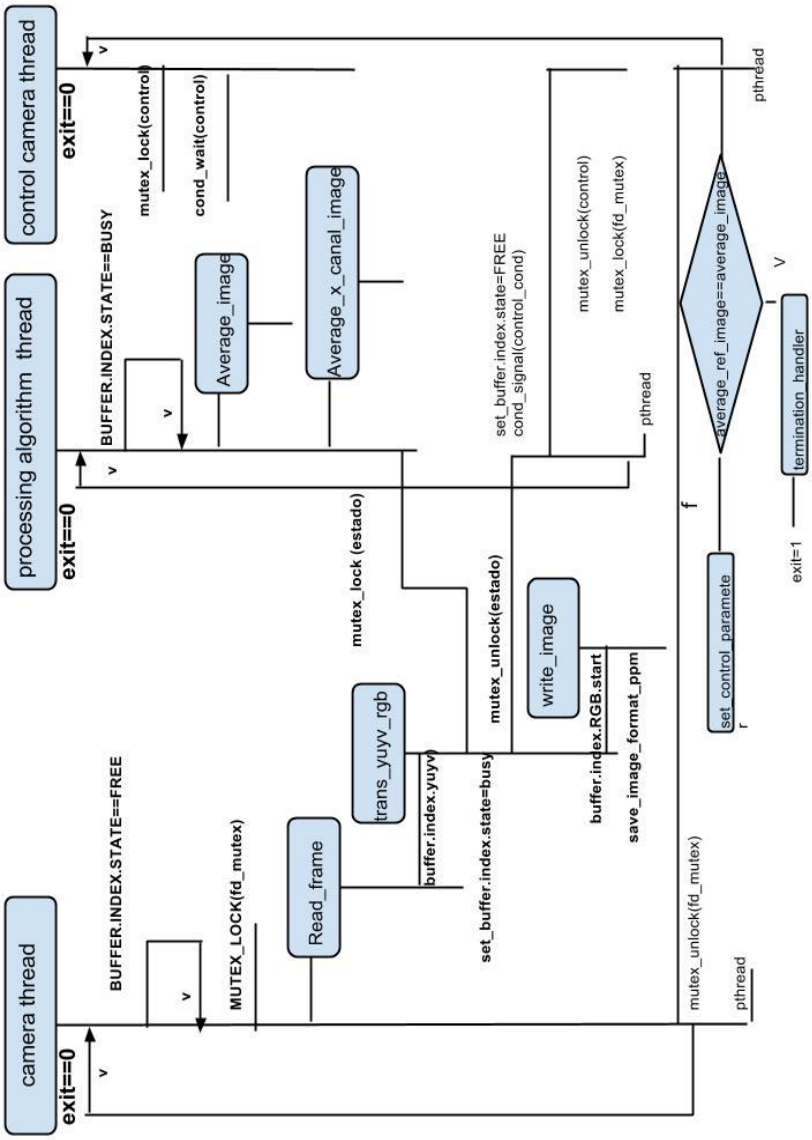
2. Realizar las capturas de las imágenes de referencia en las condiciones ambientales adecuadas.

3. Asegurarse que no exista ninguna fuente cercana que produzca alguna clase de ruido electrónico en el sensor de la cámara.

4. Revisar si el dispositivo de captura permite la configuración de los parámetros de brillo, contraste y balance de blancos.

ANEXOS

Anexo A.- Esquema general de ejecución y comunicación entre hilos del sistema



BIBLIOGRAFÍA

[1] R. F. Antonio Jesús Román Rodríguez, «Adquisición y visualización de vídeo 3D,» Escuela Ploitecnica superior de Casteldefels , Catalunya, 3 de Noviembre de 2010.

[2] J. B. A. Rivera, Segmentación de imágenes de placas vehiculares usando técnica de crecimiento de regiones, Guayaquil - Ecuador: Escuela Superior Politécnica del Litoral, 2011.

[3] P. B. G. G. G. Abraham Silberschatz, Fundamentos de Sistemas Operativos, Madrid: McGraw-Hill/Interamericana de España, S.A.U, 2006, p. 3.

[4] «Sistemas Operativos,» [En línea]. Available: www.sistemasoperativos.angelfire.com/html/2.4.2.html. [Último acceso: 29 abril 2013].

[5] «Seguridad en Unix,» 8 agosto 2003. [En línea]. Available: http://mmc.geofisica.unam.mx/LuCAS/Manuales-LuCAS/doc-unixsec/unixsec-html/node70.html. [Último acceso: 8 enero 2013].

[6] G. J., *Autocalibración y sincronización de Múltiples cámaras PTZ,* Madrid: Universidad Autónoma de Madrid, 2013.

[7] D. C. P. Pizarro, *Comparación de técnicas de calibración de cámaras digitales,* 2005.

[8] S. J. López L., *Desarrollo de un software para reconstrucción tridimensional de objetos a partir de imágenes, usando técnicas de visión artificial,* Venezuela: Universidad de Oriente Núcleo de Anzoategui, 2010.

[9] R. A. P. II, «Lectures on Image Processing.,» Vanderbilt University School of Engineering, Nashville, TN, 2008. [En línea]. Available: http://www.archive.org/details/Lectures_on_Image_Processing. [Último acceso: 30 04 2013].

[10] U. d. Málaga, *Práctica de algoritmos y programación de ordenadores,* Málaga: Departamento Lenguajes y CC. Computación.

[11] M. H. S. H. V. M. R. Bill Dirks, Video fro Linux two API Specification, copyright, 2008.

[12] H. Nanda, *Practical calibrations for a real-time digital omnidirectional camera,* Maryland: University of Maryland..

[13] J. B. A. Rivera, Segmentación de imágenes de placas vehiculares usando técnica de crecimiento de regiones, Guayaquil: Escuela Superior Politécnica del Litoral, 2011.