Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica

Implementación de una aplicación para ajustar la captura de imágenes a las condiciones de iluminación en la plataforma Leopard Board DM365

Informe de Proyecto de Graduación para optar por el título de

Ingeniera en Electrónica con el grado académico de Licenciatura

Melissa Gabriela Montero Bonilla

Cartago, 2 de mayo de 2011

Declaro que el presente Proyecto de Graduación ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos propios.

En los casos en que he utilizado bibliografía he procedido a indicar las fuentes mediante las respectivas citas bibliográficas. En consecuencia, asumo la responsabilidad total por el trabajo de graduación realizado y por el contenido del correspondiente informe final.

Melissa Gabriela Montero Bonilla

Cartago, 2 de mayo de 2011

Céd: 1-1304-0712

Instituto Tecnológico de Costa Rica
Escuela de Ingeniería Electrónica
Proyecto de Graduación
Tribunal Evaluador

Proyecto de Graduación defendido ante el presente Tribunal Evaluador como requisito para optar por el título de Ingeniera en Electrónica con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal

M.Sc. Juan Scott Chaves Noguera
Profesor Lector

Ing. Johan Carvajal Godínez
Profesor Lector

Dr. Pablo Alvarado Moya
Profesor Asesor

Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica.

Cartago, 29 de abril de 2011

Resumen

El presente proyecto tiene como objetivo implementar una aplicación para ajustar las imágenes capturadas por medio de la Leopard Board DM365 de acuerdo a las condiciones de iluminación. Además busca a través de la aplicación mostrar las capacidades que posee el procesador de video frontal de la tarjeta para la recolección de estadísticas y procesamiento de imágenes.

Con este fin, se desarrollan algoritmos de auto-balance de blancos y auto-exposición. Estos algoritmos aprovechan el módulo de estadísticas de la Leopard Board DM365 para obtener datos referentes a la imagen capturada, y a partir de ellos calcular los ajustes requeridos. Según el caso se busca eliminar la presencia de colores dominantes causados por el tipo de iluminación o ajustar la luminancia a valores considerados adecuados.

Una vez obtenidos los valores a ajustar, se utiliza una interfaz para comunicarse con el sensor de cámara o con módulos de procesamiento de la Leopard, para definir los parámetros de tiempo de exposición o ganancias.

Los algoritmos utilizados se evalúan cualitativamente y cuantitativamente. Las evaluaciones permiten la comparación de las imágenes resultantes despúes de pasar por los distintos algoritmos y la comparación de éstas con imágenes sin procesamiento.

Palabras clave: auto-balance de blancos, auto-exposición, ganancia, luminancia

Abstract

This project's objective is to implement an application capabable to adjust the acquisition of images in a Leopard Board DM365 according to the ilumination conditions. Also, this project aims to show the DM365 video processor capabilities of image processing and estimation of image statistics.

For this purpose, auto white balance and auto exposure algorithms are implemented. These algorithms adjust the captured images taking advantange of the DM365 statistics engine to compute image metrics. The auto white balance algorithms objetive is to eliminate the images color bias and the auto exposure algorithms look for the proper image luminance.

A programming interface is implemented to communicate with the camera sensor and the DM365 image processing pipeline drivers. The algorithms use this interface to define color gains and exposure time acording to the required ajustments.

Quantitative and qualitative evaluations are applied to test the implemented algorithms. The evaluations purpouse is to compare the resulting images after the algorithm application and the images without processing.

Keywords: auto white balance, auto exposure, gain, luminance

a mis queridos padres

Agradecimientos

Tomo este espacio para agradecer a la empresa y todas las personas que hicieron posible el desarrollo y término de este proyecto. Un profundo agradecimiento a la empresa RidgeRun que me dió la oportunidad de trabajar en ella y de realizar este proyecto. Dentro de la empresa, quiero agradecer especialmente al Ing. Diego Dompe por la guía, ayuda y enseñanzas brindadas. A la Ing. María Haydeé Rodríguez, aunque solo me pudo acompañar en los inicios del proyecto significó un impulso inicial indispensable, otorgándome las bases requeridas para lograr el resultado de este trabajo. Además a todos los compañeros de la empresa que de una u otra forma me dieron su apoyo y ayuda.

También quiero agredecer a mi profesor asesor, Dr Pablo Alvarado Moya, por sus consejos y enseñanzas durante el desarrollo del proyecto. Así también por la paciencia que tuvo para la revisión exhaustiva de esta tesis.

Mis más sinceros agradecimientos a mis padres Victor Montero Flores y Ana María Bonilla Piedra y a mi tía Marietta Bonilla Piedra, que han sido mi soporte incondicional no solo durante el desarrollo de este proyecto sino no durante toda mi vida. Estaré eternamente en deuda por su comprensión y dedicación. Finalmente y no por ser menos importante, quiero agradecer a mi hermano Víctor Montero Bonilla por su ayuda y paciencia.

Melissa Gabriela Montero Bonilla

Cartago, 2 de mayo de 2011

Índice general

Índice de figuras

Índice de tablas

Lista de símbolos y abreviaciones

Abreviaciones

%seg	Porcentaje de segmentación
AE	Auto exposure
AEW	Auto exposure y auto balance de blancos
AWB	Auto balance de blancos de las siglas en inglés Automatic White Balance
fps	Imágenes por segundo de las siglas en inglés frame por second
SDK	Paquete de desarrollo de software de las siglas en inglés Software Development Kit
VPBE	Sistema de procesamiento final de las siglas en inglés Video Processing Back End
VPFE	Sistema de procesamiento frontal de las siglas en inglés Video Processing Front End

Notación general

A Matriz.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix}$$

$\underline{\mathbf{x}}$ Vector.

$$\underline{\mathbf{x}} = [x_1 \; x_2 \; \ldots \; x_n]^T = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

y Escalar.

Capítulo 1

Introducción

Un sistema empotrado o embebido es aquel sistema computacional dedicado a una o pocas funciones específicas preestablecidas. Esta constituido por una combinación de software, hardware y dispositivos mecánicos que permiten el desarrollo autónomo de una aplicación. Específicamente un sistema de video empotrado es aquel dedicado a la manipulación de imágenes, como cámaras fotográficas y de video, reproductores, entre otros. Así, estos sistemas contemplan la captura, procesamiento, codificación/decodificación, almacenamiento y/o reconstrucción o despliegue de una escena. Por ejemplo, en la Figura 1.1 se muestra un sistema de video completo desde la captura hasta el despliegue o almacenamiento.

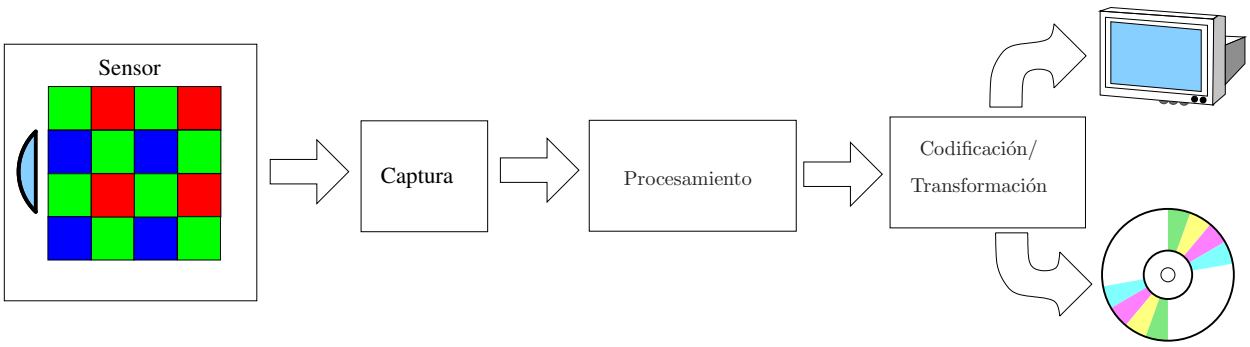


Figura 1.1: Etapas para la captura y despliegue de video

1.1 Leopard Board DM365

La *Leopard Board DM365* es una tarjeta de evaluación de hardware para el desarrollo de sistemas embebidos. Forma parte del proyecto Open Source hardware denominado Leopard Board. Fue creada para mostrar la funcionalidad del *System on Chip*(SoC) TMS320DM365 de la compañía Texas Instruments[1](TI), así como la funcionalidad de los sensores de captura

[1]Texas Instruments es una empresa internacional que se dedica al diseño de circuitos integrados y semiconductores analógicos y digitales, microprocesadores y procesadores de señales digitales

de video de la familia de módulos LI-MOD de la compañia Leopard Imaging[2].

LI-MOD consiste en una serie de módulos de cámara basados en sensores CMOS con resoluciones desde VGA, 1.3M, 2M, 3M hasta 5M.

El procesador TMS320DM365, forma parte de la familia Davinci de TI. Esta constituido por un ARM9, un subsistema de procesamiento de video (VPSS), aceleradores de video (H.264, MPEG4, MJPEG), sistemas periféricos integrados y codecs de video. El VPSS esta conformado por: una interfaz de entrada (VPFE o Video Processing Front End) para periféricos externos como sensores de cámara, decodificadores de video, entre otros, de donde se obtienen las imágenes y pasan a ser procesadas; y una interfaz de salida (VPBE o Video Processing Back End) para dispositivos de despliegue como monitores analogicos SDTV/HDTV, paneles digitales LCD, entre otros.

Debido a las características del hardware en el cual se encuentra basado la *Leopard Board DM365*, esta tarjeta provee las herramientas necesarias para el desarrollo de aplicaciones multimedia prototipo, especialmente aquellas orientadas a sistemas de video.

La *Leopard Board DM365* cuenta con soporte de software, tanto código como binarios provenientes de distintas fuentes. Así, la empresa RidgeRun ofrece un paquete de desarrollo de software (SDK), creado con el fin de simplificar el proceso de construcción de aplicaciones de software para el sistema embebido. Este SDK se encuentra basado en Linux e integra una serie de herramientas de código abierto como GStreamer, Qt, DBus.

En lo que respecta al soporte de software para sistemas de video, este paquete de desarrollo cuenta con los controladores para acceder a nivel del kernel a los módulos del hardware, tanto al VPFE como al VPBE. Además, tiene soporte en esta plataforma para el sensor de cámara de 5Mpixeles, aunque algunas funcionalidades del mismo no están completamente desarrolladas. Además, gracias a las herramientas de código abierto mencionadas anteriormente se permite realizar en el nivel de aplicación de usuario, la captura de video con ayuda del sensor de cámara, codificación/decodificación en varios formatos(MPEG4, H.264), almacenamiento y despliegue en varias salidas de video (componente, compuesta, DVI).

1.2 Limitaciones del software para Leopard Board DM365

En este paquete de software las capacidades de procesamiento de imágenes presentes en el VPFE no están siendo utilizadas. Esta es la etapa del flujo de datos de video que permitiría realizar ajustes a la imagen capturada por el sensor con el fin de mejorar la calidad según las necesidades de la aplicación. Es de especial interés para los sistemas de video que el video o imágenes capturadas sean ajustadas de acuerdo a la iluminación del escenario, pero con la ausencia del procesamiento en el software, las imágenes capturadas se ven alteradas por la iluminación, generando imágenes con colores dominantes que varían según la fuente de luz utilizada. Además, según el nivel de luz, éstas se ven muy brillantes u oscuras, generando

[2]Leopard Imaging Inc. es una compañía de alta tecnología dedicada al diseño de tarjetas con cámaras de alta definición integradas para el desarrollo de cámara fotográficas, cámaras de video, entre otros

que se pierdan detalles de la escena, especialmente de los objetos que se encuentran entre sombras o bajo una iluminación intensa.

Por tanto, con este trabajo se persigue ajustar la imagen capturada a los niveles y tipo de iluminación del entorno usando los recursos de procesamiento de la Leopard Board DM365.

1.3 Corrección de imágenes

El presente proyecto busca corregir los efectos de la iluminación, tanto la presencia de colores dominantes como de la cantidad de luz presente en la imagen para que no se vea muy clara u oscura.

La presencia de colores dominates en una escena se debe a que cada objeto refleja una radiación diferente según la fuente de luz al que es expuesto. El ojo humano es capaz de compensar automáticamente esta desviación de los colores, por una capacidad conocida como constancia de color. Así, el ojo logra que un objeto blanco sea percibido como blanco sin importar el tipo de iluminación. Sin embargo para que las imágenes capturadas por un sistema de video logren la constancia de color, es requerida una capacidad de procesamiento denominada balance de blancos.

El balance de blancos compensa las diferencias de color originadas en cambios de la iluminación. Actualmente existen gran cantidad de técnicas que llevan a cabo el balance de blancos de forma automática. En el presente proyecto se implementan dos de estos métodos utilizando las herramientas disponibles en la Leopard Board DM365.

Por otro lado, dentro del procesamiento de un sistema de video existe un ajuste denominado exposición. La exposición determina la cantidad de luz que incide en el sensor de la cámara, determinando que tan clara u oscura va a ser la imagen. Al igual que en el balance de blancos, se han desarrollado una serie de algoritmos capaces de medir el nivel de iluminación y ajustar automáticamente la exposición para obtener una iluminación adecuada en la imagen capturada, algunos de los cuales serán probados en el presente proyecto.

En la Figura 1.2 se muestra un diagrama general de la forma en que se aborda el problema. Se establece un flujo de datos de video que capture imágenes a partir del sensor de la cámara y que pasen por una etapa de procesamiento. Sobre este flujo de datos, se encuentra una interfaz de configuración y control, capaz de actuar sobre los controladores de los dispositivos encargados del manejo de video para preparar el hardware y realizar ajustes a los datos capturados. Asimismo, se incluyen dos módulos para el procesamiento de auto balance de blancos (AWB) y auto exposición (AE), los cuales se comunican con la aplicación de configuración para indicar los ajustes a los datos del video de acuerdo a sus resultados. Finalmente, un bloque de interfaz de usuario, que permite configurar las funciones automáticas AWB y/o AE y realizar las compensaciones de forma manual, actuando directamente sobre la aplicación de configuración.

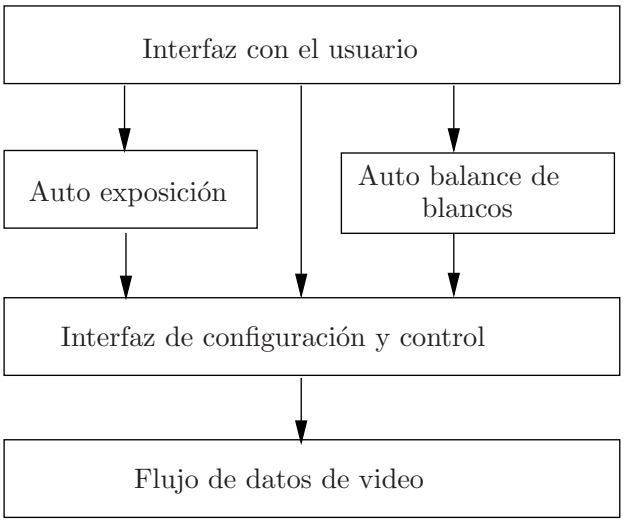


Figura 1.2: Solución propuesta

1.4 Objetivos y estructura del documento

El objetivo general del presente proyecto consiste en implementar una aplicación que permita el ajuste de la imagen capturada por la Leopard Board DM365 de acuerdo a las condiciones de iluminación y de esta manera muestre las capacidades de procesamiento de video de esta tarjeta. Con este fin, se debe desarrollar una biblioteca que implemente algoritmos de auto balance de blancos y auto exposición. Además, se debe implementar una interfaz que permita la configuración y el control del hardware de video presente en la Leopard Board DM365 como en el sensor de cámara. También, es requerido establecer un flujo de video que permita poner a disposición los datos para los ajustes deseados y muestre la funcionalidad de la aplicación. Finalmente, cuando la aplicación esté implementada se debe comparar la calidad de las imágenes procesadas y sin procesar con respecto a la constancia del color y el contraste.

Este documento se centra en explicar el desarrollo de este proyecto y los resultados obtenidos de él. El Capítulo 2 presenta los conceptos teóricos necesarios para comprender la solución. El Capítulo 3 detalla la solución propuesta en el diagrama Figura 1.2. El Capítulo 4 muestra los resultados obtenidos con la aplicación de los algoritmos de auto exposure y auto balance de blancos. Finalmente, el Capítulo 5 contiene las conclusiones del actual trabajo así como las recomendaciones para los trabajos posteriores.

Capítulo 2

Marco Teórico

2.1 Generalidades del imágenes/video digitales

Una imagen digital es una representación bidimensional de una escena, la cual es muestreada espacialmente generando una malla de pixeles. Un video digital es una secuencia de imágenes digitales, es decir se muestrea la escena en el dominio temporal y espacial. Los pixeles son unidades básicas que se utilizan para representar las caracterísitcas de la imagen, ya sea intensidad o color [9].

2.1.1 Espacios de color

Un espacio de color es un modelo utilizado para definir la composición de color en una imagen. En [9] se expresa que en principio un modelo de color es una especificación de un sistema coordenado y un subespacio dentro de este, donde cada color es representado con un punto. Generalmente se representa cada pixel por medio de tuplas. Existen variedad de espacios de color, dentro de los cuales se encuentran:

- RGB: cada color es representando es sus componentes espectrales primarias rojo (R), verde (G) y azul (B). Este modelo esta basado en el sistema de coordenadas cartesianas. Su subespacio de color es el cubo mostrado en la Figura 2.1, el cual esta limitado por RGB en tres de sus esquinas, cian, magenta y amarillo en otras tres, el origen es el color negro y blanco es la esquina más lejana al origen. En este modelo la escala de grises, es decir aquellos puntos que poseen igual magnitud de RGB, se extiende en la diagonal desde el negro hasta el blanco.
- HSI: es creado con el fin de proveer una forma más natural para el ser humano de intrepretación de las imágenes. Está conformado por tres componentes: matiz(H) que es un atributo del color que indica el tipo de color(rojo,azul o amarillo) relacionada con la frecuencia dominante del espectro de luz incidente, saturación(S) indica el grado de decoloración del color relacionado con la pureza de la frecuencia dominante y luminancia(I) que es una visión acromática de la intensidad de la luz. El subespacio

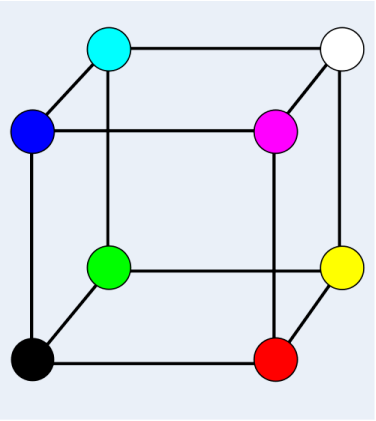


Figura 2.1: Esquema del cubo RGB [1]

de este modelo se muestra en la Figura 2.2, en el eje vertical se encuentra la luminancia, correspondiente a la escala de grises. Los colores primarios RGB se encuentran separados entre ellos por 120° y los secundarios a 60° de los primarios. El matiz de un pixel es determinado por el ángulo desde un punto de referencia, usualmente el eje rojo, y crece en contra de las manecillas de reloj desde ahí. Por su parte, la saturación es la longitud del vector desde el origen hasta el punto.

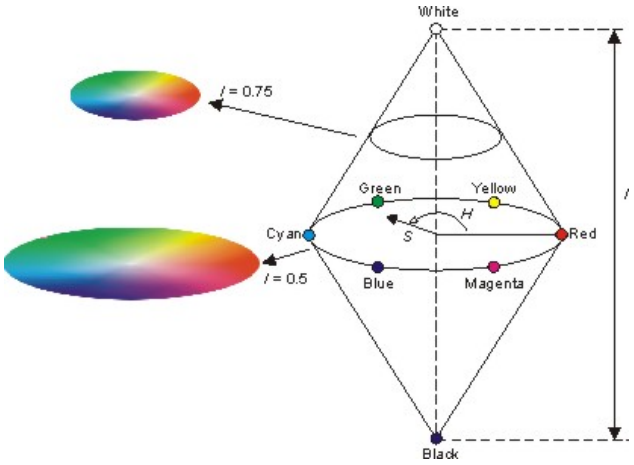


Figura 2.2: Esquema del modelo HSI [6]

- YCrCb: tiene tres componentes, Y representa la iluminación y Cb y Cr son las señales de crominancia azul y rojo respectivamente. Este espacio de color es utilizado para la compresión de imágenes y la transmisión ya que es menos redundante que el RGB.

2.1.2 Conversiones entre espacios de color HSI y RGB

Dada una imagen en formato RGB, la componente H de cada pixel es obtenida usando la ecuación:

$$H = \begin{cases} \theta & \text{si } B \leq G, \\ 360 - \theta & \text{si} B > G \end{cases} \tag{2.1}$$

donde el ángulo θ se calcula:

$$\theta = \arccos \frac{\frac{1}{2}\left[(R-G)+(R-B)\right]}{\sqrt{(R-G)^2+(R-B)(G-B)}} \tag{2.2}$$

El componente de saturación esta dado por:

$$S = 1 - \frac{3}{R+G+B}\left[min(R,G,B)\right] \tag{2.3}$$

Finalmente la luminancia se obtiene:

$$I = 1 - \frac{1}{3}(R+G+B) \tag{2.4}$$

2.1.3 Sensores de cámara

En general para adquirir imágenes digitales se utilizan módulos electrónicos de sensores de cámara que capturan las variaciones en la intensidad de la luz. Existen dos tecnologías de manufacturación de sensores: CMOS (Complementary Metal Oxide Semiconductor) y CCD (Charge Coupled Device), los cuales usan fotodetectores para convertir la luz en carga eléctrica. La diferencia entre las tecnologías se refiere a la forma en que obtienen los datos de intensidad de la luz [13].

En general, los módulos de sensores de cámara capturan la información a color utilizando un filtro Bayer RGB, el cual consiste en un arreglo de elementos foto-sensibles. Con este filtro cada pixel recibe solamente uno de los componentes de color rojo, verde o azul en lugar de todos ellos.

2.1.4 Captura de imágenes de color

De acuedo con [11], en el proceso de captura las imágenes digitales son codificadas como la intensidad de los componentes RGB. Cada uno de estos valores es afectado por tres factores físicos:

- Iluminación: el proceso de formación de imágenes inicia con una fuente de luz. Toda iluminación consiste en luz de múltiples longitudes de onda, y es representada como una distribución espectral de potencia denotada con $I(\lambda)$.
- Reflectancia: cuando las radiaciones electromagnéticas de la iluminación insiden en un objeto, estas son parcialmente absorbidas y reflejadas o transmitidas. La proporción de

reflección o transmisión varía según las longitudes de onda y las propiedades inherentes al objeto. Así, un objeto se puede caracterizar con la reflectancia o transmitancia espectral en función de las longitudes de onda. La reflectancia espectral define qué fracción de la energía incidente en el objeto va a ser reflejada y va a llegar al ojo o al sensor de la cámara y se denota con $R(\lambda)$. Así, la distribución espectral de potencia de un objeto es producto de la potencia espectral de la iluminación y de la reflectancia del objeto, y es conocida como el estímulo de color $S(\lambda)$

$$S(\lambda) = I(\lambda)R(\lambda) \tag{2.5}$$

- Sensibilidad espectral de los fotodetectores: cada filtro en el sensor de la cámara posee una sensibilidad específica para las distintas longitudes de onda. Siendo $x(\lambda)$, $y(\lambda)$ y $z(\lambda)$, las sensibilidades espectrales de los tres filtros, cuando un objeto de estímulo $S(\lambda)$ es observado, se producen tres respuestas de los tres filtros:

$$\begin{aligned} X &= \int_{400}^{700} x(\lambda)I(\lambda)R(\lambda)d\lambda \\ Y &= \int_{400}^{700} y(\lambda)I(\lambda)R(\lambda)d\lambda \\ Z &= \int_{400}^{700} z(\lambda)I(\lambda)R(\lambda)d\lambda \end{aligned} \tag{2.6}$$

2.2 Una visión general de auto balance de blancos

Como se indica en la sección 2.1.4, cuando la imagen de una escena es capturada, el valor de cada pixel va a depender de la respuesta de los tres filtros del sensor que a su vez se ven afectados por la iluminación. Según [23], el valor obtenido en cada pixel está relacionado con la temperatura del color de la fuente de luz[1]. Así, cuando un objeto blanco es iluminado bajo un color de temperatura bajo, aparece rojizo en la imagen capturada. Por el contrario, si el objeto es iluminado bajo un color de temperatura alto aparecerá azulado en la imagen capturada, es decir el color de la escena en la imagen varia según la fuente de iluminación bajo la cual fue tomada. El sistema de visión humano no logra distinguir estas diferencias de color causadas por diferentes fuentes de luz, porque cuenta con una característica denominada constancia de color. En [12] se ha postulado que los ojos son responsables de capturar las diferentes longitudes de onda de la luz reflejadas por el objeto y que el cerebro descarta la contribución de la iluminación, para que el color del objeto permanezca constante ante las diferentes iluminaciones.

Entonces, el objetivo del balance de blancos es minimizar el efecto de la iluminación $I(\lambda)$, y asegurarse que la respuestas del sensor X, Y y Z se correlacionen con la reflectancia del objeto $R(\lambda)$.

[1] Temperatura del color de una fuente de luz, se define como la temperatura a la cual la superficie de un objeto negro emitiría un espectro de luz semejante al de la fuente.

El balance de blancos utiliza algoritmos computacionales de constancia de color, para que los colores de la escena capturada permanezcan iguales sin importar la temperatura de color bajo la que se encuentren. Según [12], generalmente los algoritmos de AWB consisten en dos pasos:

- Estimación de la iluminación: puede hacerse explicitamente escogiendo la iluminación de una serie de posibilidades preestablecidas o implicitamente con supuestos de los efectos de la iluminación sobre las respuestas del sensor.

- Compensación de la imagen: generan una nueva imagen con los colores obtenidos por una iluminación estándar. Utiliza métodos como el modelo diagonal de Von Kries, que permite ajustar cada componente de color RGB individualmente. En [4] se describe este modelo como una transformación lineal, donde cada pixel de la imagen tomado bajo una iluminación desconocida $\underline{\rho}^U = (\rho_1^U, \rho_2^U, \rho_3^U)$ es mapeado al pixel correspondiente en una imagen tomada bajo una luz canónica $\underline{\rho}^C = (\rho_1^C, \rho_2^C, \rho_3^C)$, con $\underline{\rho}^C = \mathbf{M}\underline{\rho}^U$, donde $\mathbf{M}$ es una matriz diagonal. Formalmente se define:

$$\underline{\rho^{\mathbf{C}}} = \begin{bmatrix} \frac{\rho_1^C}{\rho_1^U} & 0 & 0 \\ 0 & \frac{\rho_2^C}{\rho_2^U} & 0 \\ 0 & 0 & \frac{\rho_3^C}{\rho_3^U} \end{bmatrix} \underline{\rho^{\mathbf{U}}} \tag{2.7}$$

2.3 Una visión general de auto exposición

Auto-exposición es un proceso de autocontrol para optimizar la iluminación de la imagen, ajustando la configuración del sensor. Busca controlar la exposición, término utilizado para referirse a la cantidad de luz que cae sobre el sensor durante la captura de la imagen.

Según [5] no existe una definición exacta de cual debe ser la exposición correcta, pero se puede generalizar a aquella que permita la reproducción de la regiones de interés de la escena con un nivel de iluminación aproximado a la mitad del rango disponible (rango de sensibilidad del sensor). Se habla de las regiones de interés y no del total de la escena, debido a que existen escenas con un alto contraste. Así como menciona [15] si la escena tiene un bajo contraste es posible reproducir la iluminación adecuada, pero si no, las regiones brillantes se saturan y las oscuras se enmascaran porque el rango dinámico del sensor es limitado en comparación con el rango dinámico de la escena. En especial menciona dos casos de alto contraste: contraluz y excesiva luz frontal donde la diferencia entre la iluminación del objeto(s) principal(es) es alta.

El proceso que logra el ajuste de la exposición de forma automática, según [18] implica tres pasos:

- Medición de la luz: se refiere al detector de la exposición, el cual puede ser un dispositivo separado del sensor de la imagen o el mismo sensor.

- Análisis de la escena (sistema de medición de la iluminación): se refiere al método que se va a utilizar para obtener la iluminación de la escena a partir de los datos tomados por el detector de la exposición, que puede incluir segmentación de la imagen y ponderación de regiones dependiendo del sistema. El resultado de estos métodos es utilizado para calcular la exposición adecuada.

- Compensación de la exposición: encargado de calcular y manipular los parámetros del sensor de imagen que permitan asegurar que la cantidad adecuada de luz alcance el sensor. En [16] se indica que para sensores CMOS generalmente el parámetro que permite modificar la exposición es denominado tiempo de exposición. El tiempo de exposición se define como la cantidad de tiempo (μs ó ms) que el sensor integra la luz. Así, cuando una imagen parezca estar oscura el tiempo de exposición debe aumentarse y cuando parezca clara debe disminuirse.

Además, [18] indica la existencia de dos enfoques normalmente implementados para sistemas de auto-exposición:

- *Centrado en la óptica*: un sensor externo se encarga de detectar la luz y la exposición se ajusta controlando el iris y el tiempo que el opturador se mantiene abierto.

- *Centrado en la electrónica*: se utiliza el mismo sensor de imagen para detectar la luz y la exposición se ajusta variando el tiempo de integración del sensor y la ganancia aplicada a las componentes de color.

2.4 Conociendo el VPFE del SoC DM365

VPFE, procesador de video frontal, forma parte del subsistema de video del SoC DM365, es una estructura de hardware especializada para el manejo de video y procesamiento de imágenes. A continuación se da una descripción general del VPFE tomada del manual de usuario [21]. EL VPFE esta constituido por cuatro bloques: interfaz del sensor de imagen (ISF), ducto de imagen (IPIPE), interfaz del ducto de imagen (IPIPEIF) y hardware del generador estadístico 3A (H3A).

- El ISF es responsable de recibir el video crudo (sin procesar) desde el sensor,también puede recibir video en el espacio de color YCbCr normalmente proveniente de decodificadores de video.

- El IPIPEIF es un módulo de interfaz que maneja las señales de sincronización y de datos entre ISF y el IPIPE.

- El IPIPE es un módulo de hardware programable para el procesamiento de imágenes, cuenta con una serie de etapas de procesamiento. Este módulo permite generar imágenes en los espacios de color YCbCr-4:2:2 o YCbCr-4:2:0 o imágenes crudas.

- El H3A es un módulo encargado de recolectar estádisticas acerca de los datos del video/imágenes que puedan ser utilizados por ciclos auto enfoque, auto balance de blancos o auto exposición.

Para el presente documento son de mayor interés el IPIPE y el H3A, los cuales se describen con mayor detalle a continuación.

2.4.1 IPIPE

El IPIPE solo puede ser utilizado cuando los datos de entrada se encuentran en un formato Bayer RGB. La principal tarea del IPIPE es convertir el formato Bayer a YCbCr. Está constituido por una serie de sub-bloques de procesamiento y configuración. Para el presente trabajo son de interés los módulos denominados *Balance de blancos* y *Mejoramiento de contraste y brillo*. El primero, ofrece ajuste de ganancia y de nivel para cada componente de color RGB. El ajuste de ganacia consiste en la multiplicación de los datos crudos por un coeficiente de ganacia escogido correspondiente al color. El ajuste de nivel le suma el valor indicado según el color respectivo.

Por otro lado, el sub-bloque *Mejoramiento de contraste y brillo*, se encuentra ubicado después de la conversión al espacio de color YCrCb. Éste permite aplicar ajustes al contraste y al brillo de la imagen modificando la señal Y:

$$Y_{ctrbrt} = (Y \times C) + Br \tag{2.8}$$

donde, Y es la componente de luminancia entrante en el sub-bloque, C es el factor de ajuste de contraste, Br es el factor de ajuste del brillo y Y_{ctrbrt} es la salida obtenida para la señal de iluminación.

2.4.2 H3A

El H3A cuenta con dos bloques principales: auto enfoque (AF) y auto-exposición/auto-balance de blancos (AE/AWB). El bloque AF extrae y filtra cada pixel verde de la entrada imagenes/video y provee la acumulación o los picos de los datos en una región dada.

El módulo AE/AWB, permite obtener datos estádisticos por regiones de la imagen/video capturados. Cada región corresponde a un bloque bidimensional de datos, el cual es conocido como ventana. Para cada ventana el AE/AWB puede extraer la acumulación de los valores para cada componente de color RGB y además obtener máximos y mínimos para cada componente de color o acumulación de los cuadrados.

2.5 GStreamer: API para el manejo multimedia

GStreamer es un marco de trabajo (*framework*)[2] para el desarrollo de aplicaciones multimedia. Está escrito en lenguage de programación C, usando la biblioteca GObject.

2.5.1 Conceptos básicos

En [20] se indica que los siguientes conceptos son básicos para entender el funcionamiento de GStreamer.

- Elementos: es el objeto principal de GStreamer. Cada elemento posee una función específica, puede ser leer datos desde un archivo, decodificar estos datos o enviarlos a una tarjeta de sonido. Los elementos pueden ser enlazados entre sí, y lograr que un flujo de datos pase sobre ellos. GSreamer cuenta con una serie de elementos, pero también deja abierta la posibilidad de agregar nuevos elementos con funcionalidades distintas. Existen tres tipos básicos de elementos:

  1. Elemento fuente (*source*): son los encargados de generar datos, por ejemplo leer desde un archivo o desde un controlador de cámara. No aceptan datos solo los generan.
  2. Elemento filtro (*filter*): estos reciben datos, operan sobre ellos y proveen datos nuevos en su salida. No posee un número predeterminado de entradas ni de salidas. Ejemplos de este tipo de elemento son: decodificadores/codificadores de video y audio.
  3. Elemento sumidero (*sink*): son los encargados de recibir los datos, no modifican ni generan datos, por ejemplo: reproducir audio y video.

- Contactos (*Pads*): son las entradas y salidas de los elementos, donde se pueden conectar otros elementos, existen pads fuente y sumidero. Se encargan de negociar el flujo de datos entre elementos, esta negociación se conoce como negociación de capacidades Entre sus características principales están: restringen el tipo de datos que pasan sobre ellos y para permitir un enlance entre elementos el tipo de los pads debe ser compatible.

- Cubetas (*Bins*): es un elemento contenedor, ya que incluye una colección de elementos básicos. Dado que una cubeta es del tipo elemento, se puede controlar como tal abstrayendo la complejidad de los elementos que lo componen.

- Tuberías o líneas de proceso (*Pipelines*): son cubetas de alto nivel, enlazan varios elementos y permiten la ejecución de estos. Las tuberías pueden cambiar el estado de sus elementos para lograr la ejecución de sus funcionalidades. En el momento que los elementos son creados no cuentan con la capacidad de realizar alguna acción inmediatamente, requieren un cambio de estado. GStreamer cuenta con cuatro estados:

[2] es una estructura de software, normalmente conformada por módulos concretos, por medio de los cuales otro proyecto de software puede ser organizado y desarrollado

nulo es el estado por defecto se encarga de localizar las capacidades del elemento, el estado listo donde se asignan los recursos globales, aquellos que no cambian a pesar del cambiar del flujo de datos, pausa es el estado donde se abre el flujo de datos pero no lo procesa y el estado corriendo donde se inicia el procesamiento de los datos. Existen dos maneras de ejecutar una tubería: gst-launch, es una herramienta que construye y ejecuta líneas de proceso básicas. Se emplea generalmente como herramienta de prueba de tuberías o elementos. Y por código C, usualmente las líneas de proceso se implementan en código C cuando se requiere su uso en aplicaciones.

2.6 DBus: Sistema de comunicación interprocesos

En [17] se define DBus como un sistema de comunicación interprocesos, es decir, permite la comunicación local entre procesos corriendo en el mismo sistema. Está organizado en tres capas:

- La biblioteca libdbus: provee las herramientas para permitir a dos aplicaciones conectarse entre sí e intercambiar mensajes.
- Un demonio ejecutable que funciona como bus de mensajes. Está construido sobre libdbus. Múltiples aplicaciones se pueden conectar al demonio y el demonio puede enrutar mensajes a través de estas.
- Bibliotecas adaptadas para su uso en marcos de trabajo específicos.

Todo lo que se envía y se recibe en DBus es transferido a través del bus. Hay dos tipos de buses disponibles: el bus de sesión y el bus de sistema. El primero se utiliza principalmente para la comunicación entre aplicaciones de escritorio en la misma sesión, el segundo, para la comunicación entre el sistema operativo y la sesión de escritorio, incluyendo dentro del sistema operativo al núcleo y algunos demonios o procesos.

Existen dos tipos de aplicaciones que usan DBus: servidores que escuchan por conexiones entrantes, ponen a disposición métodos, y clientes que se conectan a un servidor y hacen uso de sus métodos. Cuando una conexión es establecida, permite un flujo de mensajes a través de ellos. Cada conexión a un bus puede ser accedida en ese bus a través de un nombre único, el cual es asignado al iniciar la conexión. Además, cada servidor puede poseer varios objetos, que agrupan los métodos, estos son direccionados a través de una ruta que equivale a su nombre.

Un programa que quiera trabajar con DBus debe asegurarse que el demonio de DBus esté ejecutándose, conseguir un bus para comunicarse con el demonio de DBus, conectar con el objeto deseado utilizando su ruta. Puesto que este objeto no existe realmente dentro de la aplicación, es sólo un objeto DBus, se utiliza lo que se conoce como objeto *proxy*. Un *proxy* es una clase que enmascara los detalles de la interacción con DBus, se comporta como un objeto remoto, pero con la sintaxis propia del lenguaje de la aplicación.

En el caso de un servidor que utilice DBus se requiere crear un lazo principal para que se mantenga corriendo y conectado al bus, conectarse al bus, obtener un objeto *proxy* que represente el bus en sí mismo, registrar el nombre por el cual los clientes se conectarán y crear un objeto local que maneje los requerimientos de los clientes y definir la ruta de este objeto.

2.7 Distancia de Bhattacharyya

La distancia de Bhattacharyya según [7], es la medida de separación entre dos distribuciones normales y se define como:

$$D_b = \frac{1}{8}(\mathbf{M_2} - \mathbf{M_1})^T \left[\frac{\Sigma_1 + \Sigma_2}{2}\right]^{-1} (\mathbf{M_2} - \mathbf{M_1}) + \frac{1}{2}\ln \frac{\left|\frac{\Sigma_1+\Sigma_2}{2}\right|}{\sqrt{|\Sigma_1||\Sigma_2|}} \tag{2.9}$$

donde, M_i es el vector promedio de la distribución i y Σ_i es la matriz de covarianza de la distribución i.

El primer término indica la distancia debida a la diferencia entre los promedios de la distribución, mientras que el segundo término indica la distancia debida a las diferencias de la matriz de covarianza.

—

Capítulo 3

Software para el ajuste de los efectos de la iluminación sobre los imagen/video capturados por la Leopard Board DM365

El sistema propuesto busca poner a disposición varias opciones para la corrección de los efectos de las fuentes de luz sobre los colores e iluminación de las imágenes capturadas. Así, permite que los ajustes a las imágenes se realicen de forma automática o manual según la preferencia del usuario. Con el control manual, se tiene la capacidad de realizar acciones de configuración de hardware, actuar sobre la cadena de ajuste de datos en el sensor y controlar algunos de los módulos de procesamiento del IPIPE. Por su parte, los ajustes automáticos realizan cambios a los colores y/o exposición de la imagen en caso de ser necesario, é inician y terminan con acciones del usuario. Una vez dada la indicación de inicio se realiza la configuración de los módulos de hardware y software a utilizar y seguidamente se ingresa en un ciclo en el cual con la ayuda de los algoritmos de AWB y AE se ajustan los parámetros de la línea de procesamiento del sensor o del IPIPE según sea el caso. La Figura 3.1, muestra en forma gráfica el proceso indicado anteriormente.

Tal como se muestra en la Figura 1.2, la propuesta consta de cuatro partes principales: flujo de video, interfaz de usuario, interfaz de configuración y control del hardware y una biblioteca de auto balance de blancos y auto exposición (AEW). Cada uno de estos bloques se explican con mayor detalle en las secciones siguientes.

3.1 Flujo de video

Dado que el sistema debe corregir las imágenes capturadas por la Leopard Board DM365 se hace necesaria la presencia de un flujo de datos de video procedentes del sensor de cámara. Se utilizan las herramientas proporcionadas por el API de GStreamer para lograrlo.

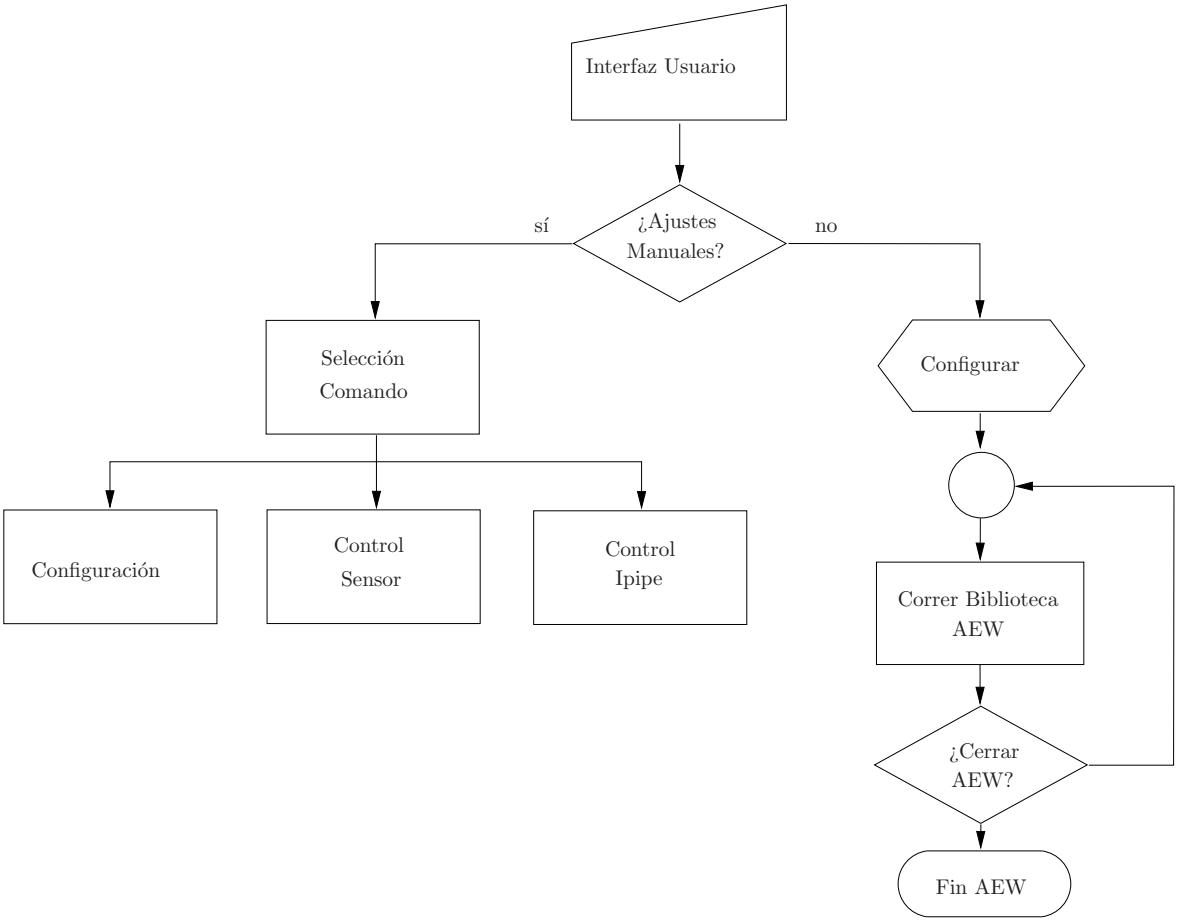


Figura 3.1: Proceso de la solución propuesta

Se deben proveer líneas de proceso de captura de video sobre las cuales van a actuar los algoritmos. Como se indicó en la sección 2.5 para tener una línea de proceso básica se necesitan tres elementos: fuente, filtro y sumidero. El elemento fuente a utilizar es *v4l2src* que se comunica directamente con el controlador del sensor de cámara y permite la captura de video. Los dos elementos restantes van a depender del formato de despliegue del video o captura de imágenes o video.

Para la comprobación y evaluación del funcionamiento de los algoritmos implementados como parte del presente proyecto se utilizan dos líneas de proceso de gstreamer. Estas líneas de proceso fueron ejecutadas con la herramienta gst-launch, ver Apéndice A.

En la Figura 3.2 se presenta la secuencia de elementos de gstreamer que conforman la línea de proceso utilizada para obtener el flujo de video requerido por los algoritmos AEW. A través del elemento v4l2src se configuran y controlan los controladores del sensor como del VPFE, los cuales permiten el inicio del flujo de datos de video. El elemento video/x-raw-yuv define las capacidades de los datos de video de v4l2src: el formato del video (UYVY), el tamaño de la imagen (640x480) y la cantidad de frames por segundo (30fps). Finalmente pasa al elemento sumidero TIDmaiVideoSink encargado del despliegue del video, se comunica con el módulo del kernel encargado manejar interfaz de salida de video compuesta.

En la Figura 3.3 se muestra la otra línea de proceso usada en este proyecto. Esta línea de

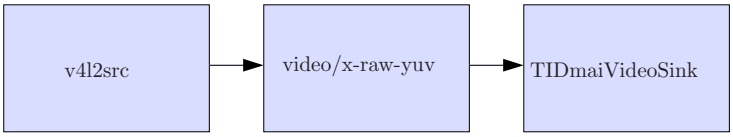


Figura 3.2: Secuencia de elementos de gstreamear para la captura y despliegue de video

proceso, se usa para obtener imágenes codificadas en formato PNG para las evaluaciones. Igualmente utiliza el elemento v4l2src como fuente y el elemento video/x-raw-yuv que define sus capacidades. Por su parte, el ffmpegcolorspace convierte un espacio de color en otro. En este caso convierte el espacio de color YUV que se obtiene de v4l2src al RGB que necesita el elemento pngenc. El elemento pngenc realiza un algoritmo de codificación para proveer datos en formato PNG. El flujo de datos llega al elemento filesink en formato PNG y este se encarga de almacenarlo en un archivo dentro del sistema local de archivos.

Figura 3.3: Secuencia de elementos de gstreamear para la captura, encodificación y almacenamiento de una imagen png

3.2 Biblioteca AEW

La biblioteca AEW es una herramienta de software desarrollada en el marco de este proyecto, que tiene como objetivo poner a disposición ajustes automáticos de balance de blancos y exposición. Consiste en una colección de rutinas implementadas en lenguaje de programación C, entre las cuáles se encuentran desarrollados algoritmos de auto exposición y auto balance de blancos, funciones que toman datos estadísticos de las imágenes (aprovechando el módulo *AEW engine* proporcionado por la Leopard), rutinas de configuración e inicialización, entre otras.

3.2.1 Organización y funcionamiento de la biblioteca

Esta biblioteca se organiza alrededor de un elemento u objeto denominado *rraew*. El elemento *rraew* consiste en una estructura de datos, donde se almacena información de las configuraciones de hardware, de los algoritmos, de funciones disponibles tanto de inicialización y ejecución de algoritmos como de control y configuración de hardware, es decir, contiene todos los datos que se desean compartir en la biblioteca.

La biblioteca AEW se organizó por componentes genéricos, con el fin de permitir la futura utilización de la biblioteca, especialmente de los algoritmos, en plataformas distintas a la Leopard Board DM365, el uso de otros sensores y la incorporación de nuevos algoritmos.

La Figura 3.4 se muestra la estructura de bloques utilizada para la organización de la información en *rraew*.

El elemento *rraew* cuenta con interfaces para interactuar con el hardware de procesamiento y captura, además de contener toda la información requirida por los algoritmos para la configuración y control del hardware. Los algoritmos de auto-balance de blancos y auto exposición necesitan comunicarse con el módulo de estadísticas para obtener los datos del flujo de video, con el sensor para ajustar los parámetros de captura tanto tiempo de exposición ó ganancias de color, además con el IPIPE para actuar en el módulo de ganancia que este ofrece como alternativa a las ganancias del sensor.

Además, *rraew* contiene un bloque con información general del video, tales como las dimensiones de la imagen, que permiten tanto a los algoritmos como a las funciones de configuración de hardware acoplarse a las características del flujo de video.

Asimismo el elemento *rraew* contiene un bloque donde se almacenan los datos estadísticos que se leen del módulo de estadísticas para que puedan ser utilizados por todos los componentes de la biblioteca.

Tiene bloques para la información de los algoritmos de auto-balance de blancos y auto-exposure. Para estos bloques el elemento *rraew* ofrece una estructura estándar que debe seguir cada algoritmo implementado: todos los algoritmos deben de proveer tres funciones básicas: inicialización, ejecución y cierre.

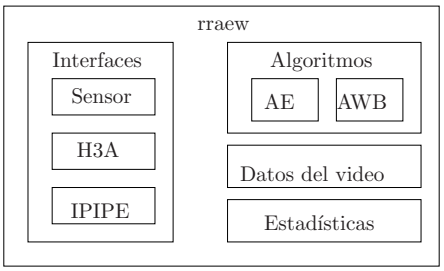


Figura 3.4: Estructura de bloques del elemento rraew

A nivel funcional la biblioteca realiza tres acciones: *crear rraew*, *destruir rraew* y *correr rraew*. Estas funciones son llamadas por la interfaz de configuración y control para aplicar auto-balance de blancos y auto-exposición sobre un flujo de video.

La función *crear rraew*, como su nombre lo indica es la encargada de crear el objeto *rraew*. Además, inicializa los módulos a utilizar y prepara los datos requeridos para la ejecución de algoritmos de AEW. En la Figura 3.5 se muestra el procedimiento que realiza la función rraew. Asigna la memoria para *rraew* y guarda en ella los datos recibidos de la interfaz de configuración y control. Estos datos son: información del video, interfaz de la plataforma, datos del sensor, tipo de ganancia y sistema de medición y algoritmos de AWB y AE escogidos. Para los algoritmos, se asignan las referencias a las funciones de inicializar, correr y cerrar correspondientes al algoritmo.

Seguidamente, esta función inicializa el módulo de procesamiento estadístico. Aunque la biblioteca trata de mantener su funcionamiento general desligado de una plataforma, para

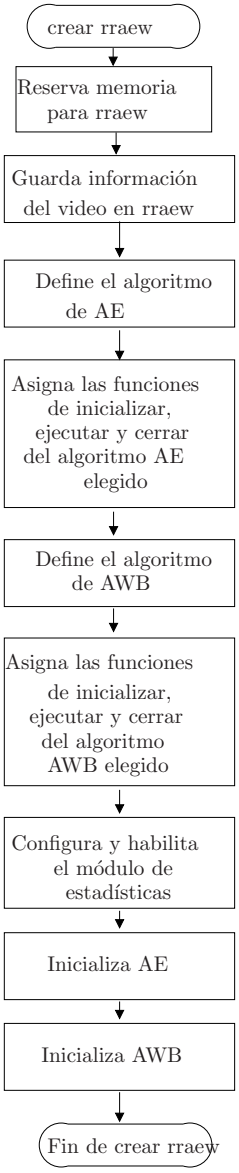


Figura 3.5: Diagrama del proceso que realiza la función crear rraew

el módulo de procesamiento estadítico se asumen ciertas características. Se espera que el módulo provea dos tipos de datos: promedios y máximos por componente de color RGB. Además, estos datos se deben proporcionar por regiones, es decir cada imagen es divida en secciones llamadas ventanas y para cada ventana se obtienen los datos estadísticos. Así la inicialización del módulo de procesamiento estadístico implica el cálculo de la cantidad de ventanas y de pixeles por ventana tanto horizontales como verticales. El criterio de elección de estos datos se basa en la idea de abarcar toda la imagen con la mayor cantidad de ventanas posibles. Sin embargo, se ofrece la posibilidad de cambiar la cantidad de ventanas a utilizar por medio de un parámetro denominado porcentaje de segmentación. El porcentaje de segmentación determina qué porcentaje del total de ventanas permitidas es el que se va a utilizar. Los datos calculados son almacenados en rraew entre los datos de configuración del módulo de estadísticas.

Con esta información se procede a configurar el módulo estadístico. Para el caso del módulo

AE/AWB del H3A de la Leopard Board DM365 el proceso de configuración se muestra en la Figura 3.6. En este proceso se define nuevamente el cálculo de parámetros. Con base en los datos de cantidad de ventanas y el tamaño de ellas se calculan parámetros específicos para este hardware mostrados en la Tabla 3.1 además de definirse otros de configuración presentados en la Tabla 3.2. El módulo AE/AWB del H3A es definido en el kernel como un dispositivo independiende con su propio controlador por lo tanto debe abrirse igual que en los casos anteriores para poder acceder a las estadísticas. Igualmente que el IPIPE este dispositivo define sus propias llamadas al sistema, AEW_S_PARAM para establecer la configuración del hardware y AEW_ENABLE para habilitarlo.

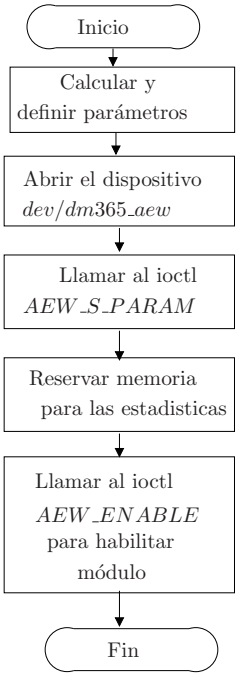


Figura 3.6: Diagrama del proceso para configurar el módulo AE/AWB del H3A

Tabla 3.1: Parámetros para definir las ventanas del módulo AE/AWB del H3A

Parámetro	**Descripción**
ancho	Ancho de la ventana (debe ser par)
alto	Alto de la ventana (debe ser par)
inicio horizontal	La posición horizontal del pixel a partir del cual se obtienen los datos
inicio vertical	La posición vertical del pixel a partir del cual se obtienen los datos
ventanas verticales	Cantidad de ventanas verticales
ventanas horizontales	Cantidad de ventanas horizontales
incremento de linea horizontal	Separación horizontal utilizada para un sub-muestreo de las ventanas
incremento de linea vertical	Separación vertical utilizada para un sub-muestreo de las ventanas

Finalmente, en *crear rraew* se prepara el camino para los auto ajustes llamando a las funciones de inicializar tanto para los algoritmos de AWB como de AE.

La función *correr rraew* es la que llama a los métodos encargados de ejecutar los auto balances, esta función corresponde a una iteración de los algoritmos elegidos. Por medio de ella se realizan tres acciones: obtener los datos estadíticos, para esto llama a la función de

Tabla 3.2: Parámetros de configuración del módulo AE/AWB del H3A

Parámetro	Descripción
habilita	Habilita o deshabilita el módulo AE/AWB
límite de saturación	Límite utilizado para definir si un pixel esta saturado o no
formato de salida	Estadísticas que se quieren obtener: solo suma, o suma y suma de cuadrados o máximos

la interfaz de la plataforma encargada de leer las estadísticas de la imagen y almacenarlas en la sección de datos estadíticos de *rraew*; ejecutar el algoritmo de auto balance de blancos usando la referencia a la función correr y de igual manera ejecutar el algoritmo de auto exposición.

La acción de leer los datos estadísticos para el módulo AE/AWB del H3A, sigue el flujo mostrado en la Figura 3.7. En caso de no encontrarse el dispositivo abierto se envía un mensaje de error porque implica que no se ha configurado el módulo de estadísticas. Si el dispositivo esta abierto se hace la llamada a la función *read*, la cual lee del dispositivo dm365_aew un buffer con los datos estadísticos de la imagen. Este buffer empaqueta las estadísticas siguiendo un formato específico para el módulo AE/AWB del H3A, por lo que se hace necesario recorrerlo y almacenarlo en el elemento *rraew* con un formato estándar definido en él.

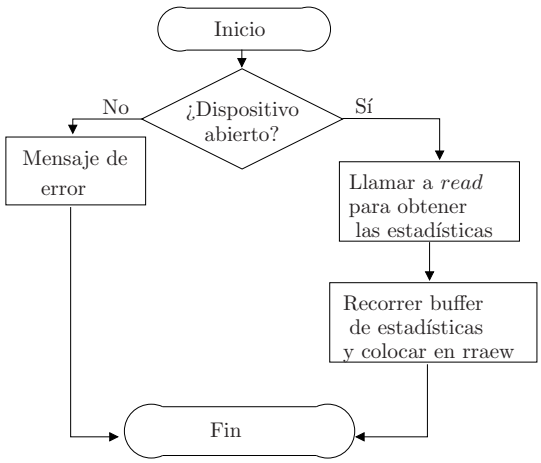


Figura 3.7: Diagrama del proceso para leer estadísticas

La función *destruir rraew*, destruye el elemento *rraew* y libera toda la memoria reservada por el mismo.

3.2.2 Algoritmos de auto balance de blancos

La biblioteca AEW implementa dos algoritmos distintos, *Mundo Gris* (*Gray World*) y *Parche Blanco* (*White Patch*) también conocido como *MaxRGB*. Además implementa una variante del *Parche Blanco*.

El primer algoritmo, *Mundo Gris*, presume que una escena típica cuenta con gran variedad de colores y que por tanto el promedio de la reflectancia de la misma es acromática (gris). Esto

implica que el promedio de la intensidad de los componentes rojo, verde y azul (RGB) deben ser iguales. Para su implementación se siguió la propuesta presentada en [11]. Dada una imagen $I(x, y)$ de tamaño $M \times N$ donde x e y indican la posición del pixel y las componentes de color rojo, verde y azul son representadas respectivamente por $I_R(x, y)$, $I_G(x, y)$ e $I_B(x, y)$, se calculan los promedios para cada color R_{avg}, G_{avg} y B_{avg} correspondientemente:

$$\begin{aligned} R_{avg} &= \frac{1}{MN}\sum_{x=1}^{M}\sum_{y=1}^{N} I_R(x, y) \\ G_{avg} &= \frac{1}{MN}\sum_{x=1}^{M}\sum_{y=1}^{N} I_G(x, y) \\ B_{avg} &= \frac{1}{MN}\sum_{x=1}^{M}\sum_{y=1}^{N} I_B(x, y) \end{aligned} \tag{3.1}$$

Si los tres promedios son iguales ya se cumple con la suposición del *Mundo Gris*. En caso contrario se deben realizar ajustes. Para ello se mantiene el componente verde sin cambio, y se define una constante de multiplicación o sea una ganancia para las dos componentes restantes:

$$\begin{aligned} \hat{\alpha} &= \frac{G_{avg}}{R_{avg}} \\ \hat{\beta} &= \frac{G_{avg}}{B_{avg}} \\ \hat{\gamma} &= \frac{G_{avg}}{G_{avg}} = 1 \end{aligned} \tag{3.2}$$

Usando los coeficientes de gananancia, se pueden ajustar los componentes RGB de cada pixel:

$$\begin{aligned} \hat{I}_R(x, y) &= \alpha I_R(x, y) \\ \hat{I}_G(x, y) &= \gamma I_G(x, y) \\ \hat{I}_B(x, y) &= \beta I_B(x, y) \end{aligned} \tag{3.3}$$

La Figura 3.8 muestra la implementación realizada del algoritmo *Mundo Gris*. Inicialmente se calcula el promedio de cada componente de color siguiendo (3.1), pero en este caso se cuenta con los promedios por ventanas de la imagen, entonces x e y representan la posición de la ventana, $M \times N$ la cantidad total de ventanas e $I_R(x, y)$, $I_G(x, y)$ e $I_B(x, y)$ corresponden a los promedios de las componentes de color RGB de cada ventana.

De igual manera si los promedios R_{avg}, G_{avg} y B_{avg} son iguales se termina el algoritmo. En caso contrario se pregunta por el tipo de ganancia escogida, ya sea del IPIPE o del sensor, con el fin de saber cuál información se debe utilizar: datos del sensor o interfaz de la plataforma. En general, los pasos a seguir son similares para ambos: primero se solicitan las ganancias anteriores y se calculan las ganancias actuales como en (3.1). Seguidamente, se encuentra el punto de diferencia entre los caminos tomados según el tipo de ganancia (donde se calculan nuevos coeficientes de multiplicación α y β). Esta diferencia se debe al punto

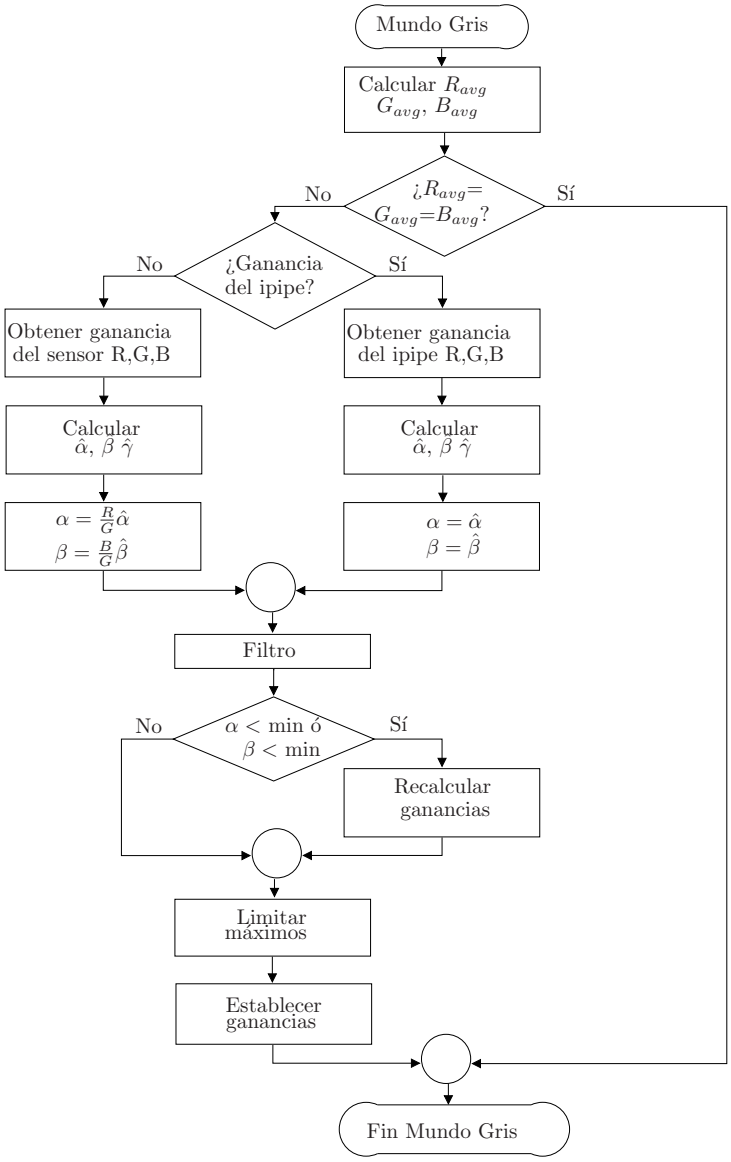


Figura 3.8: Proceso del algoritmo *Mundo Gris*

de donde se obtienen los datos estadísticos. Para el VPFE de la Leopard Board DM365 el módulo estadístico se encuentra ubicadado antes de los ajustes de ganancia del IPIPE pero después de los ajustes del sensor, entonces para el tipo de ganancia del sensor se debe tomar en cuenta el factor de multiplicación presente en el momento de tomar los datos.

Después, en la Figura 3.8 se muesta cómo los caminos se unen porque el proceso seguido es el mismo, pero se debe recordar que la información y las funciones utilizadas van a depender del tipo de ganancia. Se utiliza un filtro para evitar cambios abruptos entre iteraciones del algoritmo, el cual considera la ganancia anterior y la ganancia actual. Siendo $\mu(n)$ la entrada al filtro, $\bar{\mu(n)}$ la salida y $\bar{\mu(n-1)}$ la salida anterior se tiene:

$$\bar{\mu}(n) = a\bar{\mu}(n-1) + b\mu(n) \tag{3.4}$$

donde

$$b = 1 - a,\ 0 < a < 1 \tag{3.5}$$

Luego se verifican que las ganancias obtenidas se encuentren dentro del rango de ganancia permitido. Si se encuentran por debajo del mínimo se recalculan las ganancias para que la ganancia menor se encuentre en el límite inferior del rango. Si sobrepasan el máximo se recortan al límite superior del rango.

Finalmente se fijan las ganancias en el hardware correspondiente, utilizando la referencia a la función de control respectiva.

El segundo algoritmo *Parche Blanco*, asume que la máxima respuesta en una imagen es causada por un reflector perfecto, por lo tanto representa el color de la iluminación. Estas suposiciones se basan en la teoría Retinex que sostiene para el sistema visual humano la percepción de blanco es asociada con la máxima señal de los conos. Por lo tanto, el objetivo del algoritmo es igualar los máximos de cada componente de color RGB. De igual manera, para la implementación se siguió la propuesta de [11]. Se obtienen los máximos para cada color R_{max}, G_{max} y B_{max}:

$$\begin{aligned} R_{max} &= \max(I_R(x,y)) \\ G_{max} &= \max(I_G(x,y)) \\ B_{max} &= \max(I_B(x,y)) \end{aligned} \tag{3.6}$$

En caso que los tres máximos no sean iguales, se mantiene el componente verde sin cambio, y se define una constante de multiplicación o sea una ganancia para las dos componentes restantes:

$$\begin{aligned} \tilde{\alpha} &= \frac{G_{max}}{R_{max}} \\ \tilde{\beta} &= \frac{G_{max}}{B_{max}} \\ \tilde{\gamma} &= \frac{G_{max}}{G_{max}} = 1 \end{aligned} \tag{3.7}$$

Igualmente, usando los coeficientes de gananancia, se pueden ajustar los componentes RGB de cada pixel, como en (3.3). La implementación de este algoritmo tiene un proceso similar al presentado en la Figura 3.8, con la diferencia que en lugar de obtener los promedios se tienen los maximos R_{max}, G_{max} y B_{max} y se utiliza (3.7) para calcular los coeficientes de multiplicación $\tilde{\alpha}, \tilde{\beta}, \tilde{\gamma}$ en lugar de $\hat{\alpha}, \hat{\beta}, \hat{\gamma}$.

El último algoritmo de AWB utilizado, es una variante del ya presentado *Parche Blanco*. El *Parche Blanco* como se presentó anteriormente es considerado en [8] como una implementación ingenua, ya que se encuentra a la merced de perturbaciones en los cálculos, debido a la presencia de algunos pixeles brillantes erróneos causados, por ejemplo, por ruido, además de la saturación de los valores de alta intensidad por los recortes en los bits de datos. Por tanto, [8] indica la necesidad de un preprocesamiento de los datos. Tomando en consideración la propuesta de [11] de dividir la imagen, se calcula el promedio de los máximos de las ventanas de la imagen en lugar de utilizar los máximos absolutos, donde se tiene una imagen divida en $M \times N$ ventanas. El máximo de las componentes de color rojo, verde y azul por ventana son representadas por $I_{Rmax}(x,y)$, $I_{Gmax}(x,y)$ e $I_{Bmax}(x,y)$ y x e y indican la posición de la

ventana:

$$R_{Avgmax} = \frac{1}{MN}\sum_{x=1}^{M}\sum_{y=1}^{N} I_{Rmax}(x,y)$$
$$G_{Avgmax} = \frac{1}{MN}\sum_{x=1}^{M}\sum_{y=1}^{N} I_{Gmax}(x,y) \tag{3.8}$$
$$B_{Avgmax} = \frac{1}{MN}\sum_{x=1}^{M}\sum_{y=1}^{N} I_{Bmax}(x,y)$$

Para todos los algoritmos implementados las funciones de inicialización y cierre requeridas por el elemento *rraew* no realizan ninguna acción, ya que los algoritmos no lo necesitan.

3.2.3 Algoritmos de auto exposición

Para el AE el sistema utiliza un enfoque centrado en la electrónica, basado en la idea del tono medio. Se utiliza la idea presentada en [22], para calcular el tiempo de exposición. Parten de la expresión:

$$Bl = kLGT(F/\sharp)^{-2} \tag{3.9}$$

La cual relaciona el valor de iluminación con los factores de exposición, donde Bl es el nivel de iluminación de la imagen capturada, k es una constante, L es la iluminación del ambiente, G es el control de ganancia automático, $F/\sharp$ es el valor de apertura y T el tiempo de integración.

Bl_n y T_n denotan la iluminación y el tiempo de exposición de la imagen actual, respectivamente, y Bl_{opt} y T_{opt} la iluminación y el tiempo de exposición de la imagen con la exposición óptima. Además, se asume que para dos imágenes continuas en una secuencia G y L permanecen sin cambio y que la apertura para un sensor CMOS generalmente se encuentra prefijada a su valor máximo. Con estas variables se deriva entonces:

$$T_{opt} = T_n \cdot Bl_{opt}/Bl_n \tag{3.10}$$

Como se basa en la idea del tono medio la exposición adecuada se obtiene en la mitad del rango disponible. Se define Bl_{opt} en 128 ya que se utiliza un sistema de 8 bits. Además se establece un rango de iluminación óptimo, para evitar constantes ajustes y lograr la estabilización del sistema, se adopta el rango [100,130] utilizado en [22].

Para calcular Bl_n, se implementan seis sistemas de medición de la iluminación:

- Promedio: Bl_n de la imagen es obtenida promediando la información de iluminación de todos los pixeles de la imagen.
- *Spot*: se utiliza la información de una región pequeña en el centro de la imagen, 4% del área total de la misma, el resto es ignorada. Así, Bl_n se obtiene con el promedio de la iluminación de los pixeles en esta área del centro.

- *Partial*: este método mide una región más amplia que *Spot*, usa un 10% del área total de la imágen. Bl_n se obtiene de igual manera que con *Spot*.
- *Center weighted metering*: utiliza los datos del toda la imagen, pero como su nombre lo indica posee un sistema de ponderación . Así, divide la imagen en dos regiones una central de aproximadamente 15% del total del área de la imagen y el fondo con el restante 85%. En el calculo de Bl_n se le da un peso del 75% de sensibilidad a la iluminación de la región central y el restante 25% a la iluminación del fondo.
- *Segmented*: también es conocido como el método matriz, divide la imagen en varias secciones y después combina la iluminación obtenida en los diferentes puntos para calcular Bl_n. Especifícamente se implementa el método de segmentación presentado en [10]. Este método busca lograr una adecuada iluminación del objeto principal en una escena a contraluz. El método propuesto divide la imagen en 5 regiones como se muestra en la Figura 3.9.

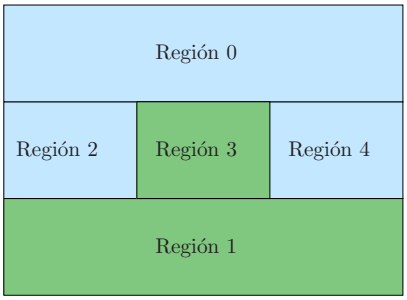


Figura 3.9: Segmentación de la imagen para AE

Asume que el fondo de la escena se ubica en la parte superior de la imagen y el objeto principal en el centro de la misma. La iluminación de cada región es ponderada de acuerdo al nivel de contraluz, el cual se obtiene con un factor de diferencia de iluminación(D_b) entre el objeto principal y el fondo, se calcula usando (3.11).

$$D_b = [R_0 + max(R_2, R_3)](R_1 + R_4) \tag{3.11}$$

donde R_i denota la iluminación promedio de la Región i, según la Figura 3.9. Con el factor D_b se obtiene un peso normalizado entre [0,5, 1] N_b según

$$N_b = \begin{cases} 0,5 & \text{si } D_b < T_1, \\ \frac{0,5B_b}{T_2-T_1} + \frac{0.5T_2-T_1}{T_2-T_1} & \text{si} T_1 < D_b < T_2, \\ 1 & \text{si } D_b > T_2 \end{cases} \tag{3.12}$$

donde T_1 y T_2 son valores umbral para definir si la iluminación es normal, a contraluz o con excesivo contraste por contraluz. Se utiliza N_b como el factor de peso para la iluminación del objeto principal y $1 - N_b$ para el fondo. Así, a la iluminación del objeto principal, regiones 1 y 4, se le da mayor peso entre mayor sea D_b

Para el algoritmo de AE implementado la función de inicialización, define el sistema de medición a utilizar y calcula los paramétros para delimitar las regiones utilizadas en él. La función de cerrar no realiza ninguna acción.

3.3 Interfaz de configuración y control

La interfaz de configuración y control está implementada como un servidor de D-Bus, el cual se ejecuta como un demonio (*daemon*), es decir en un segundo plano por lo que no es controlado directamente por el usuario. Esta interfaz es la encargada de proporcionar las funciones para comunicarse con los controladores del sensor de cámara, la interfaz de procesamiento de video de la plataforma y la biblioteca AEW, es decir permite la interconexión de los elementos del sistema. Para la comunicación con los controladores tanto del sensor como de la plataforma se utilizan *ioctl*, llamadas de sistema.

Esta interfaz se puede dividir en tres grupos funcionales:

- Control sobre el sensor.
- Control sobre el IPIPE.
- Configuración.

El grupo de control sobre el sensor incluye todas aquellas funciones que actúen sobre algún módulo de la cadena de ajustes del sensor. Se implementaron solamente los ajustes que permiten corregir los efectos de la iluminación tópico del presente proyecto. La Figura 3.10 muestra un esquema con los métodos usados, los cuales se describen a continuación.

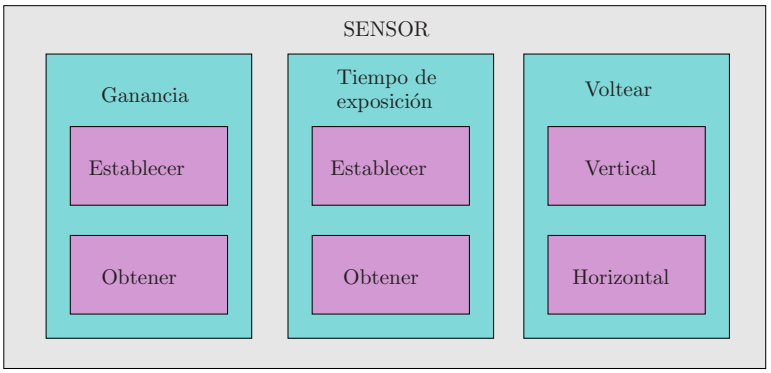


Figura 3.10: Funciones de control sobre el sensor

Los dispositivos usualmente poseen una serie de controles configurables por el usuario, los cuales varían de un dispositivo a otro. Por este motivo, dentro de la API de v4l2, se definen estándares para controles *ioctl*, los cuales proporcionan la información y los mecanismos para crear una interfaz de comunicación con los diferentes controles de los dispositivos. Cada control es accedido usando una identificación (ID). V4L2 define varios ID para propósitos específicos, pero los controladores pueden definir los propios. Además cada control posee atributos tales como valores por defecto y rangos de valores permitidos que permiten su caracterización [19].

En el sistema operativo GNU/Linux los dispositivos son accedidos mediante archivos especiales estadarizados. Para poder comunicarse con ellos se requiere abrir el dispositivo, con lo que se obtiene un descriptor de archivo correspondiente a este. Para abrir un dispositivo se utiliza la ruta del mismo, en el caso de dispositivos de captura de video se utiliza el archivo "/dev/video0" [19].

Todos los bloques para establecer un control del sensor siguen el mismo procedimiento mostrado en la Figura 3.11. El primer paso en el proceso de establecer las ganancias en el sensor es verificar si se cuenta con un descriptor de archivo para el sensor de cámara, en caso de no tenerlo se procede a abrir el sensor.

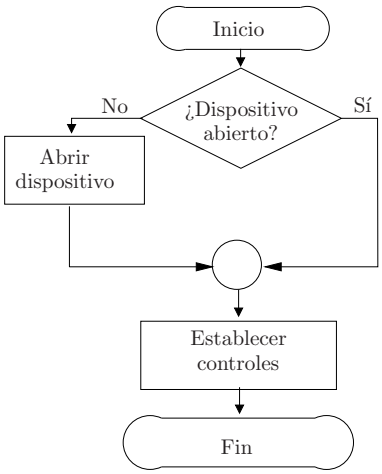


Figura 3.11: Proceso de las funciones para establecer controles del sensor

Con el sensor abierto se procede a establecer el valor de cada control que se desea ajustar. El proceso de establecer un control del sensor se muestra en la Figura 3.12. En una estructura especializada de V4l2 para controles de dispositivos se define el ID del control a establecer. Después se asigna el valor para la ganancia con un formato definido por el controlador del sensor mt9p031. Finalmente se utiliza la llamada al sistema VIDIOC_S_CTRL, la cual se comunica con el sensor de camára y establece el control indicado.

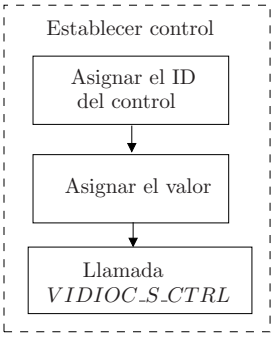


Figura 3.12: Proceso para establecer el valor de un control ioctl

Igualmente, todos los bloques para obtener el valor de los controles del sensor realizan el mismo proceso mostrado en la Figura 3.13.

Los pasos a seguir para obtener el valor de las ganancias se muestra en la Figura 3.14. Se utiliza el ioctl VIDIOC_G_CTRL para obtener cada ganancias. El valor obtenido tiene un formato definido por el controlador del sensor, este debe ser convertido al formato de datos del solicitante del control ya sea la interfaz de usuario o la biblioteca. Los valores obtenidos se envían al solitante.

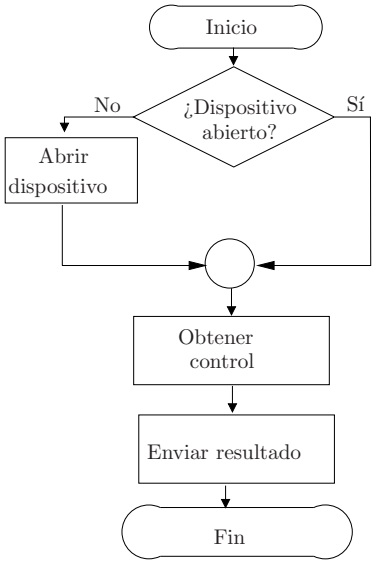


Figura 3.13: Proceso de la función para obtener los controles del sensor

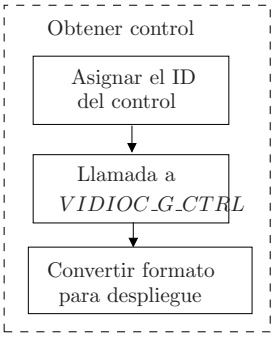


Figura 3.14: Proceso para obtener el valor de un control ioctl

El módulo para establecer la ganacia del sensor, obtiene los valores de cada componente de color RGB ya sea desde la interfaz de usuario o de la biblioteca AEW y los convierte al formato requerido por el controlador del mt9p031. Por otro lado, el módulo para obtener la ganancia del sensor realiza la funcionalidad opuesta obtiene los datos del sensor y los convierte a la forma requerida para la interfaz del usuario o de la biblioteca.

Para las ganancias el controlador del sensor mt9p031 usa los controles:

- V4L2_CID_RED_BALANCE para la ganancia roja.
- V4L2_CID_BLUE_BALANCE para la ganancia azul.
- V4L2_CID_AUTOGAIN para una de las ganancias verde.
- V4L2_CID_BRIGHTNESS para la otra ganancia verde.

Las ganancias tienen un formato de punto fijo Q22,10.

Continuando con las funciones, el bloque para establecer el tiempo de exposición representa la función encargada de fijar el tiempo de integración de la luz en el sensor. El ID del control de exposure usado es V4L2_CID_EXPOSURE, y el valor es escrito en punto fijo Q24,8. Por su parte, la función de obtener el tiempo de exposición adquiere el tiempo establecido en el momento de la solicitud, siguiendo el mismo procedimiento que para obtener las ganancias.

Además, para el sensor se habilitaron las funciones de voltear las imágenes tanto vertical como horizontalmente. Como sus nombres lo indican estas funciones invierten los datos de las imágenes, las columnas o las filas según corresponda. Para el caso particular del sensor mt9p031, se debe considerar que al girar la imagen se invierte el orden del patrón Bayer, es decir si en una fila sin invertir los componentes se envian en orden RGRG en una fila invertida se envian GRGR. En la Figura 3.15 se muestra el proceso para voltear las imágenes. Se establece el valor del control de acuerdo a si se desea rotar la imagen o si se desea mantener en su posición por defecto. Para establecer la rotación se sigue el proceso de establecer un control mostrado en la Figura 3.12. El ID usado para la rotación vertical es V4L2_CID_VFLIP y para la rotación horizontal V4L2_CID_HFLIP.

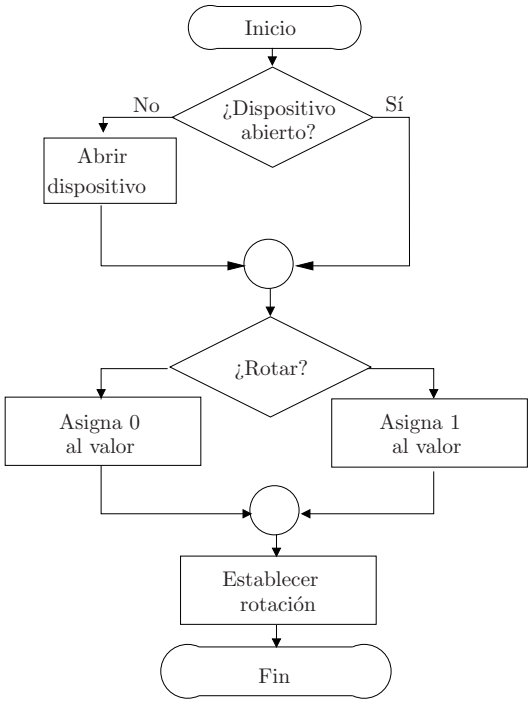


Figura 3.15: Proceso de las funciones para voltear las imágenes

El grupo de control sobre el IPIPE está constituido por todas aquellas funciones que realizan algún ajuste sobre el flujo de video usando módulos del IPIPE. La Figura 3.16 muestra las funciones implementadas para el IPIPE de la Leopard Board DM365.

El control sobre los paramétros de los módulos del IPIPE se realizan a través de llamadas al sistema ioctl proporcionadas por el controlador del IPIPE. En la Figura 3.17 se muestra el proceso que se debe seguir para controlar los parámetros de uno de los módulos del IPIPE. El dispositivo a controlar debe encontrarse abierto para poder comunicarse con él. Para el caso del IPIPE la ruta del dispositivo a utilizar es "/dev/davinci_previewer". Si se encuentra abierto, se convierten los parámetros al formato requerido por el controlador del dispositivo. Cada módulo posee una estructura definida donde se deben almacenar los parámetros que lo configuran en un formato específico. Cada módulo se identifica con un ID por lo cual se debe asignar la ID del módulo sobre el cual se desea actuar. Se definen los valores de los datos, asignando la estructura correspondiente a los parámetros del módulo. Y finalmente se llama a PREV_S_PARAM encargada de establecer los parámetros al hardware.

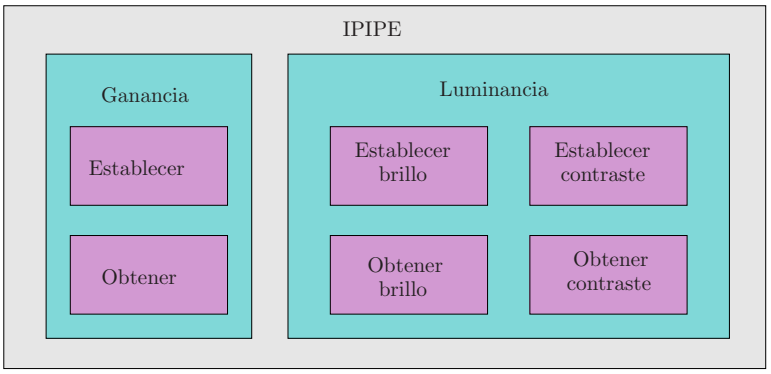


Figura 3.16: Funciones de control sobre el IPIPE

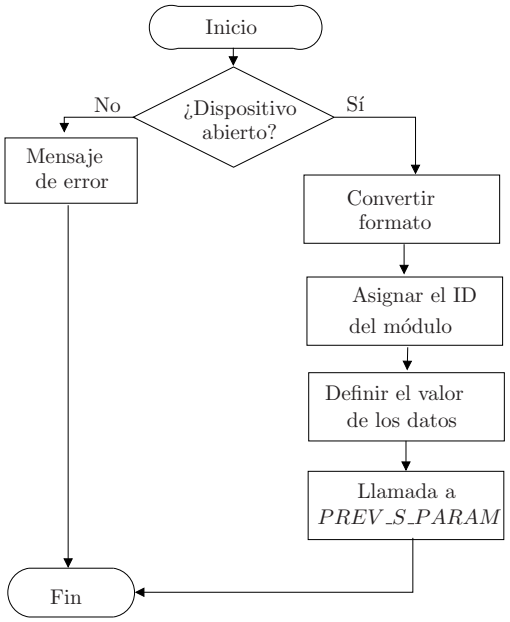


Figura 3.17: Proceso para establecer los parámetros de un módulo del IPIPE

En la Figura 3.18 se muestra el proceso para obtener los parámetros de un módulo del IPIPE. Para obtener los paramétros se debe crear una estrutura vacía correspodiente al módulo. Con la llamada ioctl PREV_G_PARAM, se llena esta estructura con los parámetros actuales en el módulo. Los datos obtenidos son convertidos para utilizar formato de punto flotante y enviados al cliente para su despliegue.

De la Figura 3.16, el primer bloque representa las funciones de ganancia digital, que son aquellas que actuan sobre el módulo "Balance de blancos". La estructura de parámetros para este módulo, contiene para cada componente de color R, Gr, Gb y B: el nivel de compensación y la ganancia constituida por un valor entero y uno decimal. De ésta, solo se utilizan los coeficientes de multiplicación de ganancias dejando los niveles de compensación (offset) fijados en cero. Así las funciones permiten establecer u obtener el factor de multiplicación de cada componente de color RGB, conviertiendo los formatos según se requiera entre el controlador del IPIPE, la biblioteca AEW y la interfaz de usuario.

También se utiliza el módulo del IPIPE para el ajuste de la luminacia, el cual actua sobre la

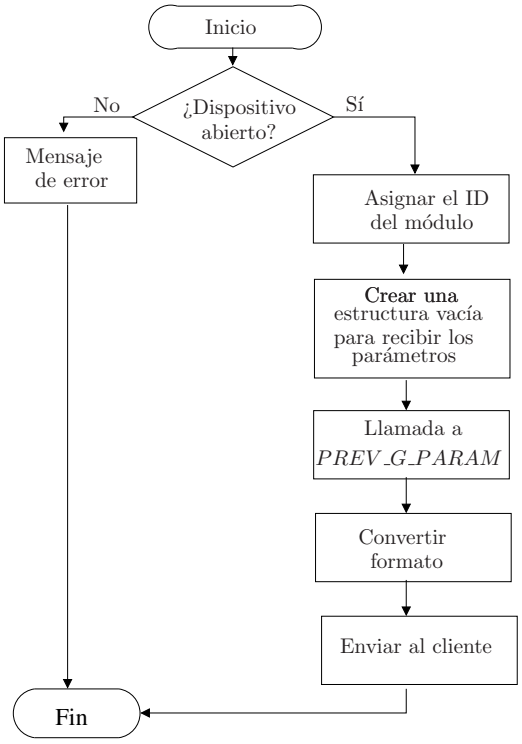


Figura 3.18: Proceso para obtener los parámetros de un módulo del IPIPE

componente de iluminación del espacio de color YUV. La estructura de parámetros correspondiente da acceso a sus dos ajustes: un nivel de compensación del brillo y un coeficiente de multiplicación denominado contraste. El brillo tiene un formato entero de 8 bits y el contraste un formato de punto fijo Q4,4. Igualmente se implementan dos funciones: una para establecer los valores y la otra para obtenerlos.

Finalmente el grupo de configuración, reúne aquellas funciones que fijan los parámetros que van a determinar el comportamiento del sistema y permiten la inicialización de los módulos correspondientes. Estas están únicamente disponibles desde la interfaz de usuario. Los bloques implementados para esta categoría se muestran en la Figura 3.19.

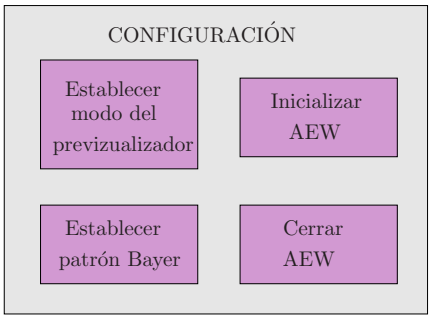


Figura 3.19: Funciones de configuración

La función de establecer el modo de previsualización es necesaria para trabajar con el IPIPE. El IPIPE por defecto no forma parte del flujo de video porque se encuentra deshabilitado, por lo que se debe establecer un modo de operación para habilitarlo. En la Figura 3.20 se muestra el proceso seguido por la función de establecer el modo de previsualización:

configura el previsualizador en modo continuo o un solo cuadro mediante la llamada al sistema PREV_S_OPER_MODE. Además configura el previsualizador con los valores por defecto usando otra llamada al sistema PREV_S_CONFIG con una estructura de configuración nula.

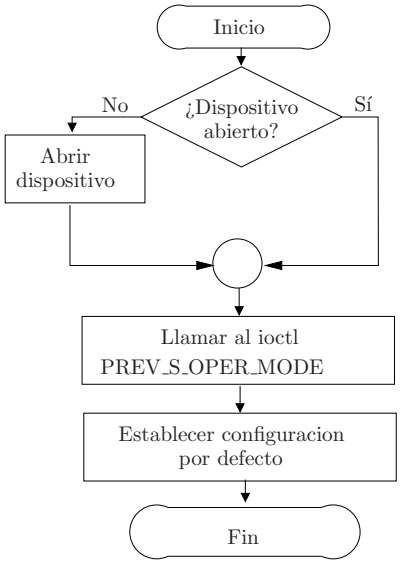


Figura 3.20: Proceso para establecer el modo de previsualización

El bloque de establecer el patrón Bayer, utiliza la capacidad que tiene el VPFE para definir el patrón Bayer en el cual le van a llegar los datos desde el sensor de la cámara, es decir la posición de cada componente de color RGrBGb que esta relacionada con las rotaciones sobre los datos del sensor. Esta configuración se hace mediante la llamada al sistema PREV_S_CONFIG, pero en esta ocasión la estructura de configuración se modifica para cambiar el patrón Bayer. En estructura de configuración se indica cual es la posición de cada componente de color RGrBGb.

Para utilizar los ajustes automáticos presentes en la biblioteca AEW, la interfaz de configuración pone a disposición las funciones de inicializar y cerrar AEW.

La función de inicializar configura la biblioteca escogiendo:

- El algoritmo de auto balance de blancos.
- El algoritmo de auto exposición y su sistema de medición.
- El tipo de ganancia.
- La mínima cantidad de imágenes por segundo.
- El porcentaje de segmentación.

Además proporciona las capacidades y funciones tanto del sensor y como del IPIPE de la Leopard Board DM365. Esto se logra con una llamada a la función crear rraew. El proceso que realiza esta función se muestra en la Figura 3.21

La función de cerrar el AEW se encarga de terminar el hilo creado para el ciclo de AEW y llamar a cerrar rraew.

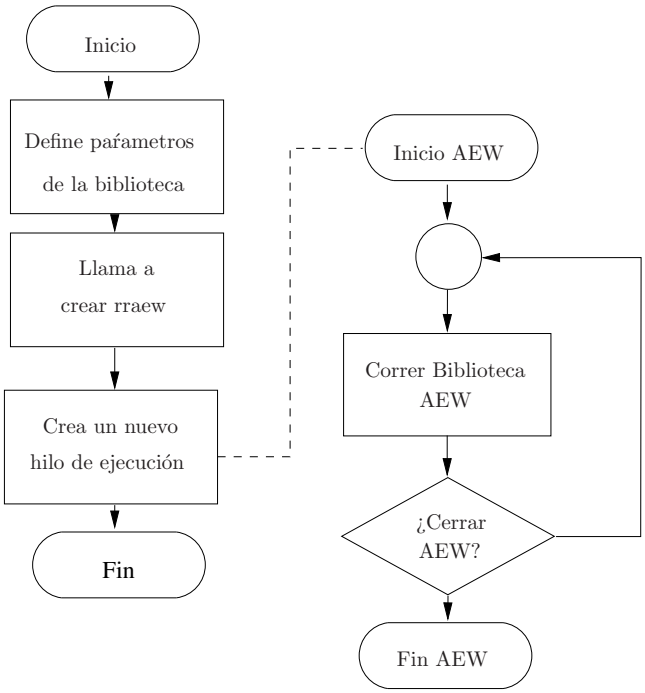


Figura 3.21: Diagrama de la función para inicializar AEW

3.4 Interfaz de usuario

La interfaz de usuario consiste en una aplicación estilo consola, donde se acceden a sus funcionalidades a través de comandos. Es implementada como un cliente de D-Bus, el cual se comunica a través de D-Bus con la interfaz de configuración y control. Pone a disposición los ajustes y configuraciones de esta interfaz.

Capítulo 4

Resultados y Análisis

En el presente capítulo se muestran y analizan los resultados de la evaluaciones realizadas a los algoritmos de la aplicación implementada. Se realiza una evaluación cualitativa de las correcciones realizadas a las imágenes tanto para el AWB como para el AE. Además, para los algoritmos de AWB se utiliza una evaluación cuantitativa, que permite medir la distancia de las imágenes procesadas con respecto a una imagen de referencia. Finalmente se caracteriza el uso del CPU de la aplicación en varias configuraciones.

4.1 Algoritmos a prueba

La evaluación del proyecto consiste en probar los algoritmos de AWB y AE implementados sobre imágenes capturadas por la Leopard Board DM365. En el caso del AWB se evalúan los tres tipos de algoritmos presentados *Mundo Gris*, *Parche Blanco* y la variante del *Parche Blanco*, con las ganancias tanto del IPIPE como del sensor y con variaciones en el porcentaje de segmentación. En la Tabla 4.1 se listan los algoritmos de WB evaluados, el tipo de ganancia usada y el porcentaje de segmentanción con su respectiva abreviación.

Tabla 4.1: Abreviaciones de los algoritmos de auto balance de blancos utilizados para las pruebas

Abreviación	**Algoritmo**	**Tipo de ganancia**	**Segmentación %**
GW-ipipe	Gray World	IPIPE	100
GW-sensor	Gray World	Sensor	100
WP-ipipe	White Patch	IPIPE	100
WP-sensor	White Patch	Sensor	100
WP2_100-ipipe	Variante del White Patch	IPIPE	100
WP2_100-sensor	Variante del White Patch	Sensor	50
WP2_50-ipipe	Variante del White Patch	IPIPE	100
WP2_50-sensor	Variante del White Patch	Sensor	50

Igualmente para el AE se evalúan los algoritmos implementados. En la Tabla 4.2 se enu-

meran los algoritmos de AE y sus abreviaciones.

Tabla 4.2: Abreviaciones de los algoritmos de auto exposición utilizados para las pruebas

Abreviación	Sistema de medición
Prom	Promedio
C-W	Center weighted
Seg	Segmented
P	Partial
S	Spot

En adelante en este documento, los algoritmos serán nombrados con las abreviaciones de las Tablas 4.1 y 4.2. Además el término "Ninguno" se utiliza para las imágenes capturadas sin aplicarles procesamiento de balance de blancos o exposición.

Los algoritmos de AE y AWB modifican dos parámetros para las imágenes a capturar: la exposición y la ganancia. Específicamente, para el sensor utilizado se modifica un registro denominado ancho de obturador (SW) que tiene una relación proporcional al tiempo de exposición (ver Apéndice C). Con el fin de observar el efecto que pueden causar estos ajustes a la luminancia de la imagen y la relación que existe entre ellos se obtuvo la Figura 4.1. En ella se grafica la luminancia promedio de una escena en relación a variaciones tanto del SW como de la ganancia global (GG). La escena consistió en una superficie lisa y se trató de mantener una iluminación constante e uniforme durante las pruebas. El rango de variación del SW es de 0 a 2495, mientras que el rango de la ganancia es de 1 a 4,5. La luminancia se obtuvo en 10 bits por lo que el máximo valor que puede obtener es 1023. De la Figura 4.1 se verifica que al aumentar el SW correspondiente al tiempo de exposición y la ganancia se aumenta la luminancia de la imagen. Además se evidencia que a mayor ganancia la luminancia llega a su punto de saturación con un menor valor de exposición. También se observa de las curvas obtenidas para las ganancias inferiores a 2 tienen una forma y magnitud similar, indicando que variaciones pequeñas de ganancia no afectan en gran medida la luminancia de la imagen. Basándose en este resultado y la observación de que los ajustes de ganancia realizados por los algoritmos de AWB generalmente son pequeños, la evaluación de los algoritmos de AE y AWB se realizan independientemente. Se asume que los ajustes realizados por AWB no afectan las mediciones de iluminación para los algoritmos de AE.

4.2 Evaluación cuantitativa de los algoritmos de AWB

La evaluación cuantitativa realizada consiste en la comparación de los algoritmos de AWB implementados usando un grupo de imágenes con diferentes escenas e iluminaciones. Las imágenes utilizadas para la evaluación forman parte de un conjunto de imágenes calibradas disponibles en [14]. Específicamente se utilizaron tres de las escenas del grupo de mínimas especularidades: papel1, macbeth y munsell4 (Figura 4.2). De las las once iluminaciones a

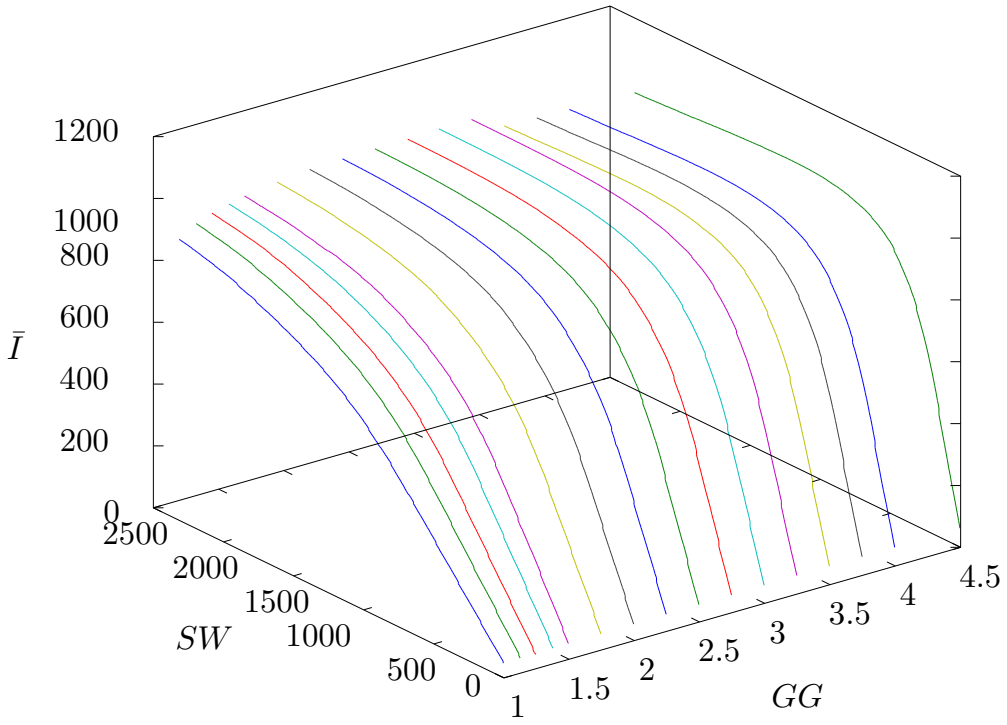


Figura 4.1: Efectos del shutter width y de la ganancia global del sensor sobre la luminancia de la imagen

las cuales fueron tomadas las escenas se escogieron las imágenes de seis de ellas citadas en la Tabla 4.3 con la abreviación a utilizar para refererenciarlas.

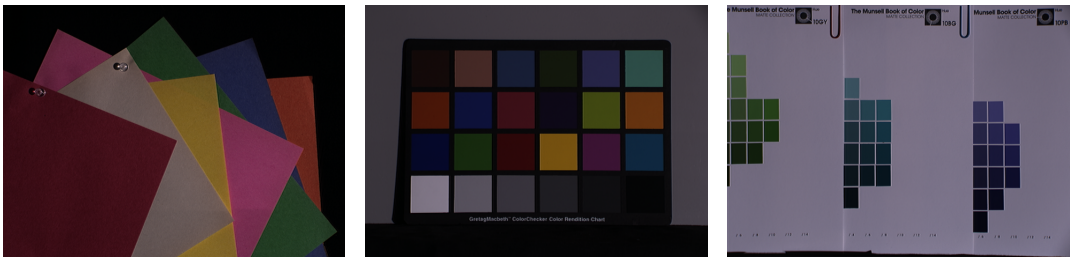
Tabla 4.3: Iluminaciones de las imágenes utilizadas para evaluación cuantitativa

Abreviación	**Iluminación**
syl-50MR16Q	Sylvania 50MR16Q (12VDC) (A basic tungsten bulb)
syl-50MR16Q+	Sylvania 50MR16Q (12VDC) + Roscolux 3202 Full Blue filter
solux-3500K	Solux 3500K (12VDC) (Emulation of daylight)
solux-3500K+	Solux 3500K (12VDC) + Roscolux 3202
ph-ulm	Philips Ultralume Fluorescent (110VAC)
syl-wwf	Sylvania Warm White Fluorescent (110VAC)

Las imágenes del conjunto de evaluación citadas anteriormente son usadas como la escena para la capturar con la Leopard Board DM365. A cada escena se le aplican todos los algoritmos de AWB y se obtienen las imágenes a analizar. Para la captura de estas imágenes se despliegan las imágenes escenas en un monitor LCD, y se coloca delante de él la Leopard, de manera que la cámara logre visualizar por completo la pantalla. Tanto el monitor como la tarjeta se ubican dentro de un espacio cerrado que impide la entrada de iluminación ajena a la proporcionada por el LCD. Para cada imagen se ajustó el valor de exposición para evitar en lo posible regiones de saturación o de oscuridad.

Cada imagen capturada es analizada en el espacio de color HSI, particularmente en el plano HS para considerar solamente los datos correspondientes al color. Para el análisis de cada

imagen se obtiene la medida de dispersión de los colores de los pixeles obtiendo la matriz de covarianza de $\rho = S\cos(2\pi H)$ y de $\zeta = S\sin(2\pi H)$. Además, se calculan los valores promedios de ρ y ζ tomando en cuenta todos los pixeles. Una vez obtenidos estos datos para todas las imágenes es posible calcular la relación de similitud entre dos imágenes para el plano HS por medio de la distancia de Bhattacharyya. Para la evaluación las imágenes capturadas son comparadas con una imagen de referencia correspondiente a la imagen de escena con la iluminación syl-50MR16Q.



(a) (b) (c)

Figura 4.2: Imágenes de referencia para la evaluación cuantitativa de los algoritmos de AWB: (a) papel, (b) macbeth, (c) munsell

En la Tabla 4.4 se muestran los cálculos de las distancias de Bhattacharyya para las imágenes munsell. De ella se advierte que las imágenes sin ningún algoritmo con mayor distacia a la imagen referencia son las que utilizan el filtro. Además, de las imágenes a las cuales se le realiza procesamiento se observa, el que logra una menor distancia es el algoritmo GW-sensor para la iluminación syl-50MR16Q+, 0.040. Por otra parte, se evidencia que para todas las iluminaciones el algoritmo WP alcanza las distancias mayores. Además para la mayoría de las imágenes los algoritmos de AWB disminuyen la distancia a la imagen referencia en comparación con la imagen sin procesamiento, a excepción del algoritmo WP-sensor para la iluminación syl-50MR16Q y la iluminación solux-3500K y en general los resultados para la iluminación syl-wwf.

Tabla 4.4: Distancia de Bachatarya para la imagen munsell

Iluminación \Algoritmo	**GW**		**WP**		**WP2_100**		**WP2_50**		**Ninguno**
	sensor	**ipipe**	**sensor**	**ipipe**	**sensor**	**ipipe**	**sensor**	**ipipe**	
syl-50MR16Q	0,228	0,259	0,391	0,353	0,229	0,263	0,298	0,272	0,390
syl-50MR16Q+	0,040	0,136	0,678	0,399	0,046	0,140	0,038	0,131	1,953
solux-3500K	0,221	0,219	0,534	0,452	0,223	0,225	0,218	0,229	0,533
solux-3500K+	0,072	0,155	0,603	0,155	0,071	0,152	0,072	0,151	2,621
ph-ulm	0,157	0,159	0,157	0,200	0,157	0,164	0,161	0,154	0,741
syl-wwf	0,227	0,226	0,226	0,224	0,223	0,227	0,221	0,230	0,223

Para las imágenes macbeth en la Tabla 4.5 se muestran las distancias de Bhattacharyya con respecto a la imagen de referencia. Igual que en el caso anterior los algoritmos de AWB logran disminuir la distancia a la referencia, a excepción de las imágenes con las iluminaciones syl-50MR16Q, solux-3500K+, ph-ulm y syl-wwf a las cuales se les aplica el algoritmo WP-ipipe. Para este grupo de imágenes el algoritmo que logra la menor distancia es WP2_50-ipipe para la iluminación syl-50MR16Q.

Tabla 4.5: Distancia de Bhattacharyya para la imagen macbeth

Iluminación\Algoritmo	GW		WP		WP2_100		WP2_50		Ninguno
	sensor	ipipe	sensor	ipipe	sensor	ipipe	sensor	ipipe	
syl-50MR16Q	0,060	0,023	0,078	0,218	0,127	0,207	0,126	0,012	0,209
syl-50MR16Q+	0,064	0,212	0,301	0,443	0,068	0,226	0,085	0,229	0,722
solux-3500K	0,112	0,225	0,437	0,459	0,110	0,244	0,156	0,265	0,769
solux-3500K+	0,128	0,186	0,065	0,316	0,126	0,195	0,127	0,202	0,267
ph-ulm	0,033	0,043	0,326	0,438	0,034	0,039	0,033	0,054	0,437
syl-wwf	0,300	0,166	0,304	0,490	0,294	0,176	0,297	0,213	0,339

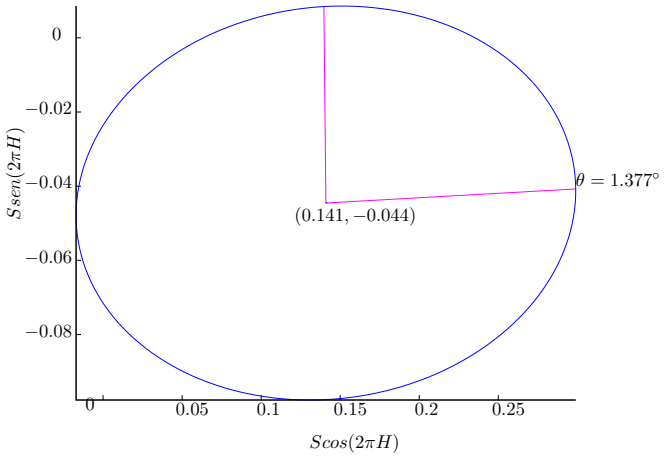
Para la última imagen evaluada, la del papel, se muestra el cálculo de las distancias de Bhattacharyya en la Tabla 4.6. En esta Tabla se ven mayor cantidad de casos (además del algoritmo WP) donde la imagen sin procesamiento obtiene una distancia menor, para las iluminaciones ph-ulm y syl-wwf cuando se utiliza ganancia en el ipipe. Asimismo, se evidencia que para los algoritmos menos el WP con la iluminación syl-50MR16Q+, una menor distancia se obtiene al utilizar la ganancia del sensor. Por su parte, en este grupo de imágenes el algoritmo que logra la menor distancia es el WP-sensor, con 0,020. Este resultado se debe a que esta escena no contiene regiones saturadas por lo cual el supuesto del cual parte el algoritmo WP se cumple.

Tabla 4.6: Distancia de Bachatarya para la imagen papel

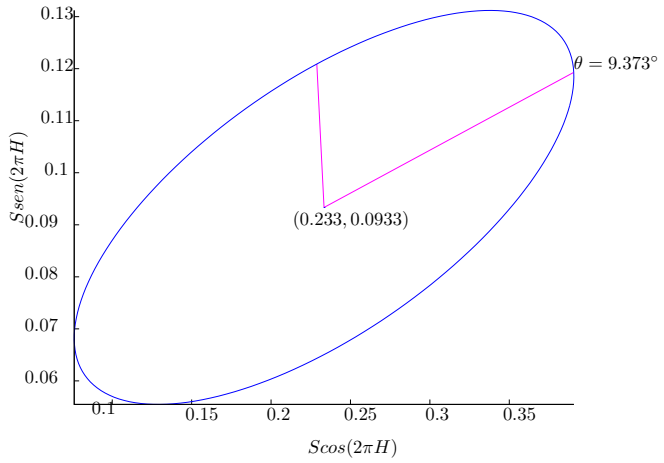
Iluminación\Algoritmo	GW		WP		WP2_100		WP2_50		Ninguno
	sensor	ipipe	sensor	ipipe	sensor	ipipe	sensor	ipipe	
syl-50MR16Q	0,066	0,108	0,051	0,182	0,065	0,107	0,059	0,110	0,114
syl-50MR16Q+	0,061	0,159	0,176	0,075	0,053	0,172	0,053	0,175	0,202
solux-3500K	0,045	0,056	0,030	0,135	0,045	0,055	0,045	0,055	0,081
solux-3500K+	0,040	0,188	0,089	0,215	0,039	0,194	0,040	0,214	0,383
ph-ulm	0,048	0,217	0,046	0,283	0,032	0,225	0,031	0,242	0,114
syl-wwf	0,052	0,215	0,020	0,485	0,050	0,234	0,049	0,259	0,089

Para ejemplificar los ajustes realizados por el balance de blancos, se obtienen las gráficas de la Figura 4.3. En ellos se muestra la distribución de los pixeles en el plano HS para varias imágenes de la escena del papel con la iluminación syl-wwf. Por medio de la elipse se delimita la mayor concentración de valores obtenidos por los pixeles, centrada en su promedio. Para obtener la elipse se calcularon los valores propios y vectores propios de la matriz de covarianza. Los vectores propios corresponden a los ejes principales de la elipse y los valores propios las logitudes de estos.

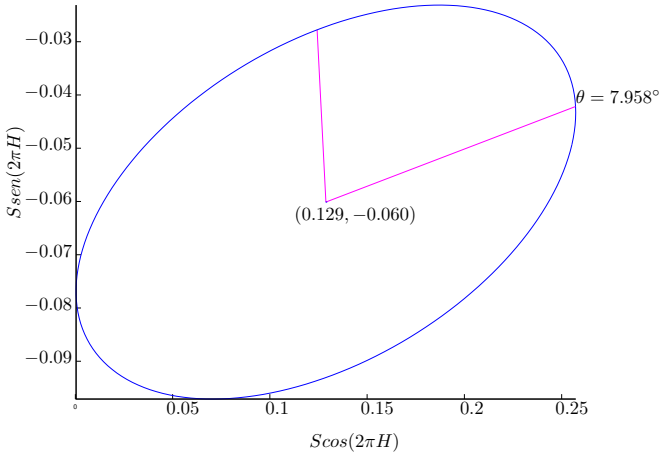
A fin de comparar los efectos del AWB, se presesenta la distribución de la imagen referencia y la distribución de las imágenes capturadas por la Leopard sin procesar y con el algoritmo WP-sensor, el cual obtuvo la menor distancia de Bhattacharyya. La distribución de referencia como se ve en la Figura 4.3a, tiene la media en saturación $S = 0,148$ y tono $H = -17,33°$ y el ángulo del eje principal de la elipse es de $1,377°$. Por su parte la imagen capturada sin procesamiento Figura 4.3b, tiene la media en saturación $S = 0,168$ y tono $H = 14,35°$ y el ángulo del eje principal de la elipse es de $9,373°$, un error de saturación de 14,35% y de tono de $31,68°$ para la media. La imagen obtenida con el algoritmo WP-sensor, cuya distribución



(a)



(b)



(c)

Figura 4.3: Ejemplos de la distribución de pixeles en el plano HS de la escena del papel: (a) para la imagen de referencia, (b) para la imagen sin AWB, (c) para la imagen procesada con el algoritmo WP-sensor

se muestra en la Figura 4.3c ajusta la media de la distribución para acercarse a la media de la referencia, con saturación $S = 0,153$ y tono $H = -24,94°$, disminuyendo el error de saturación a 3,47% y de tono a $7,94°$. Por su parte en las Figuras se observa claramente cómo se corrige la distribución, siendo la forma de la elipse de WP más redondeada como la de la referencia. Y además reduce el ángulo del eje principal de la elipse, de $9,373°$(sin procesar) a $7,958°$(con WP-sensor) acercándose al ángulo de referencia.

4.3 Evaluación cualitativa de los algoritmos

Según [3] tanto los algoritmos de auto balance de blancos como los de auto exposición se basan en modelos de procesos subjetivos dependientes de la escena, la cámara y la percepción de la persona. Se indica que por tanto, la forma más certera de evaluar su desempeño es realizando pruebas cualitativas, las cuales se basan en la práctica de usar seres humanos para realizar mediciones (psicometría).

Así, la evaluación de los algoritmos de AWB y AE implementados para la Leopard Board DM365 se lleva a cabo mediante una evaluación cualitativa. Se utilizan dos de las técnicas de medición subjetiva presentadas en [3]: escalado por categorías y clasificación jerárquica (ranking). En el escalado por categorías se utilizan una serie de categorías etiquetadas con una secuencia lógica, las imágenes son juzgadas basadas en un atributo específico dentro de una de las categorías propuestas. Al igual que en [15] se definieron cinco categorías, como se muestran en la Tabla 4.7. Por su parte, la clasificación jerárquica busca ordenar las imágenes presentadas de la mejor a la peor ó de la peor a la mejor de acuerdo a un atributo específico, se le atribuye un peso de acuerdo a su posición, siendo uno el peso de la peor imagen.

Tabla 4.7: Calificaciones de las categorías

Calificación	Categoría
5	Excelente
4	Muy Bueno
3	Bueno
2	Regular
1	Malo

Los algoritmos de AWB y AE se evalúan de manera independiente, por lo cual se obtiene un juego de imágenes de evaluación para cada tipo de procesamiento. En ambos casos las imágenes fueron capturadas con la Leopard Board DM365 y corresponden a escenas típicas: retratos, paisajes e interiores (oficina). En la evaluación de algoritmos de AWB y AE se tomaron cuatro escenas para cada procesamiento. En la Figura 4.4 se ilustran las escenas usadas para evaluar los algoritmos de AWB y en la Figura 4.5 las escenas para los algoritmos de AE.

Para cada escena seleccionada, se obtiene una imagen corregida por cada algoritmo a prueba y la imagen sin procesamiento. Para el conjunto de imágenes de AWB se mantuvo el tiempo

(a) **(b)**

(c) **(d)**

Figura 4.4: Escenas para la evaluación de los algoritmos de AWB: (a) Paisaje, (b) Pizarra, (c) Muñecos y (d) Retrato

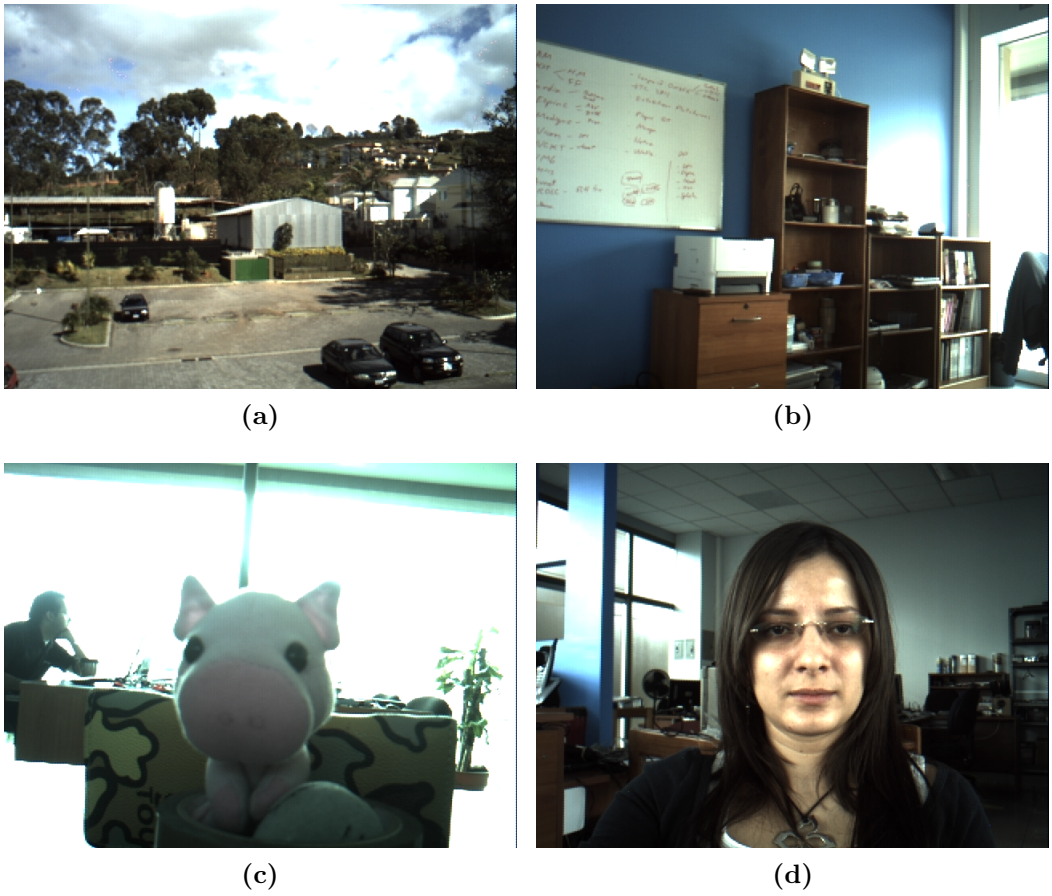
de exposición constante en 18,3 ms. En el caso del juego de imágenes de AE se inicializan las ganancias del sensor con el algoritmo *Mundo Gris* y se mantienen para todas las tomas de la respectiva escena.

La evaluación de las imágenes se realizó mediante una aplicación programada en la plataforma ASP.net con el lenguaje C. La misma está ligada a un sistema de base de datos en la que se almacena la información de las imágenes de evaluación, los grupos en los que se encuentran agrupadas y las votaciones realizadas por los observadores. Cada grupo corresponde a una de las escenas capturadas.

Al iniciar la aplicación, al observador se le presentan las instrucciones para realizar la evaluación de imágenes. En el Apéndice B están las indicaciones utilizadas. Una vez leídas las instrucciones, el observador procede a evaluar cada imagen. Las imágenes son mostradas por grupos (escenas) en una interfaz como la Figura 4.6

La calificación que cada observador asigna a cada imagen como la posición en la que fue evaluada dentro de un grupo es almacenada en las votaciones de la base de datos.

Las evaluación fue aplicada a 12 personas familiarizadas con la escena real. Todos los observadores realizaron la prueba en una misma computadora portátil, bajo las mismas



(a) **(b)**

(c) **(d)**

Figura 4.5: Escenas para la evaluación de los algoritmos de AE: (a) Paisaje, (b) Pizarra, (c) Chancho y (d) Retrato

condiciones en el entorno.

En [3] se indica que la evaluación subjetiva de un estímulo visual, se puede describir como una distribución estadística tomada de un grupo de observadores, los cuales permiten describir a un observador estándar y que se caracteriza por una distribución normal de las respuestas alrededor de la media. Así, para cada imagen se calcula la media de las calificaciones y de las posiciones, la desviación estándar y el error relativo de las observaciones.

4.3.1 Evaluación de los algoritmos de AWB

Para la primera imagen de la evaluación cualitativa de AWB, la escena de la pizarra, se obtienen los resultados de la Tabla 4.8. Para esta escena el algoritmo que obtuvo la mayor calificación y se considera logra el mejor balance de blancos es WP2_50-ipipe. Por el contrario, la imagen que se considera posee el peor balance de blancos es aquella que no posee ningún procesamiento. Sin embargo, la imagen obtenida con el procesamiento WP-sensor posee la misma calificación que Ninguno y con WP-ipipe una calificación con una diferencia de 0,250, por tanto estos tres ajustes se pueden considerar son los que rinden de peor manera, siendo calificados entre regular y malos. La escena evaluada al ser capturada por la cámara

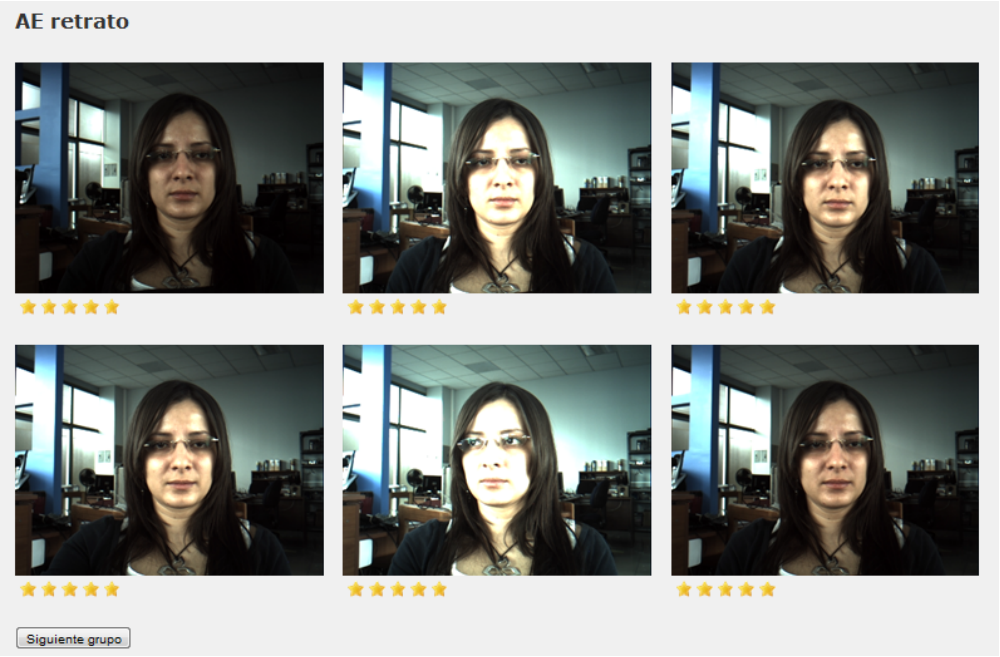


Figura 4.6: Interfaz ejemplo de un grupo de imágenes

posee regiones saturadas lo que entorpece la suposición del algoritmo *Parche Blanco*, siendo que el máximo valor no corresponde a la iluminación de la escena. Asismismo se observa que la variante del *Parche Blanco* logra sobrellevar este inconveniente rindiendo como el mejor algoritmo de balance de blancos para la escena.

Tabla 4.8: Calificación de los algoritmos de AWB para la escena de la pizarra

Algoritmo	Calificación			Posición		
	Promedio	Desviación Estándar	Error relativo	Promedio	Desviación Estándar	Error relativo
Ninguno	1,750	0,622	35,519	2,000	0,953	47,67
GW-ipipe	3,833	1,337	34,881	6,833	2,250	32,92
GW-sensor	3,500	1,000	28,571	5,083	2,193	43,15
WP-ipipe	2,000	0,739	36,927	2,417	1,240	51,31
WP-sensor	1,750	0,754	43,073	2,167	0,835	38,53
WP2_50-ipipe	4,417	0,793	17,954	7,250	1,658	22,87
WP2_50-sensor	3,583	0,996	27,801	5,750	1,658	28,84
WP2_100-ipipe	3,917	0,996	25,435	7,083	1,782	25,15
WP2_100-sensor	3,833	0,718	18,724	6,417	1,084	16,89

Los resultados obtenidos de la evaluación cualitativa para la escena de los muñecos se muestran en la Tabla 4.9. En ella se observa que el algoritmo GW tanto con ganancia en el sensor como en el ipipe fue seleccionado como el mejor balance de blancos, logrando una calificación de cuatro correspondiente a *Muy Bueno*. Esta imagen presenta mayor diversidad de colores que la anterior logrando que la suposición del GW se cumpla y logre el mejor rendimiento. Por su parte el algoritmo WP2_50-ipipe alcanza la misma calificación. Al igual que la escena anterior la imagen sin procesamiento obtiene el último lugar en el posicionamiento jerárquico, con una diferencia de 0,333 con respecto al algoritmo de *Parche Blanco* . La baja calificación de este algoritmo de igual manera que en la escena anterior se debe a la presencia de píxeles saturados en la imagen.

Tabla 4.9: Calificación de los algoritmos de AWB para la escena de los muñecos

Algoritmo	Calificación			Posición		
	Promedio	Desviación Estándar	Error relativo	Promedio	Desviación Estándar	Error relativo
Ninguno	1,667	0,651	39,080	2,000	1,809	90,45
GW-ipipe	4,000	0,739	18,464	7,250	0,965	13,31
GW-sensor	4,000	0,953	23,837	7,083	1,621	22,89
WP-ipipe	2,000	0,739	36,927	2,667	0,888	33,29
WP-sensor	2,000	0,426	21,320	2,500	1,000	40
WP2_50-ipipe	4,000	1,279	31,980	7,167	2,368	33,04
WP2_50-sensor	3,083	0,515	16,700	4,917	0,996	20,26
WP2_100-ipipe	3,583	0,996	27,801	6,250	2,221	35,53
WP2_100-sensor	3,083	1,240	40,220	5,167	2,368	45,83

En la Tabla 4.10 se recopilan las calificaciones y posiciones promedio de los votos de la evaluación cualitativa para el paisaje. En este caso el mejor algoritmo tanto por la calificación como la posición en el WP2_100-sensor y el peor lo ocupa el WP-ipipe. Cabe destacar que los algoritmo WP2_100-ipipe, WP2_50-sensor, WP2_50-ipipe, GW-ipipe tienen un rendimiento similar, con diferencias de calificación no mayores a 0,25 y estando en la categoría de *Bueno* a *Muy Bueno*.

Tabla 4.10: Calificación de los algoritmos de AWB para la escena del paisaje

Algoritmo	Calificación			Posición		
	Promedio	Desviación Estándar	Error relativo	Promedio	Desviación Estándar	Error relativo
Ninguno	1,667	0,778	46,710	2,333	0,985	42,2
GW-ipipe	3,500	1,087	31,060	6,083	2,065	33,95
GW-sensor	3,250	1,215	37,398	5,833	2,038	34,93
WP-ipipe	1,583	0,793	50,082	2,000	1,128	56,41
WP-sensor	1,917	0,900	46,958	2,583	1,730	66,96
WP2_50-ipipe	3,417	0,996	29,157	6,167	1,467	23,79
WP2_50-sensor	3,667	0,985	26,856	6,500	2,111	32,47
WP2_100-ipipe	3,417	0,996	29,157	6,583	1,730	26,28
WP2_100-sensor	3,917	1,379	35,207	6,917	2,353	34,02

Para la escena del retrato tanto las puntuaciones como las posiciones de los algoritmos evaluados se presentan en la Tabla 4.11. El algoritmo con la mejor puntuación y mayor posición corresponde a WP2_100-ipipe, seguido por el algoritmo WP2_50-ipipe con una diferencia de calificación de 0,250 y por el algoritmo GW-ipipe con una diferencia de 0,417. Estos algoritmos son los que logran reproducir el color de la piel de forma más acertada. Por su parte, los algoritmos con las posiciones y calificaciones más bajas son: Ninguno, WP-ipipe y WP-sensor respectivamente de peor a mejor. Es notable para esta escena que la imagen sin algoritmo de AWB obtiene una puntuación de 1,167, inferior a las anteriores por 0,5 ó más, indicando que los observadores notan una mayor desviación de colores sobre la piel de una persona.

En la Figura 4.7 se muestran gráficamente las calificaciones obtenidas por cada algoritmo para cada una de las escenas utilizadas en la evaluación. Seguido del nombre del algoritmo, se indica entre paréntesis el símbolo utilizado para los puntos y el color. De este gráfico, se ve claramente una separación de los algoritmos, aquellos que tienen bajas puntuaciones inferiores a los 2.5 y los de mayores calificaciones superiores a 3. Entre el grupo de calificaciones bajas se encuentran Ninguno, WP-ipipe y WP-sensor. El que posee las calificaciones más

Tabla 4.11: Calificación de los algoritmos de AWB para la escena del retrato

Algoritmo	Calificación			Posición		
	Promedio	Desviación Estándar	Error relativo	Promedio	Desviación Estándar	Error relativo
Ninguno	1,167	0,389	33,364	1,083	0,289	26,65
GW-ipipe	3,833	1,030	26,866	7,250	1,765	24,34
GW-sensor	3,167	0,835	26,364	5,417	1,505	27,79
WP-ipipe	1,583	0,515	32,522	2,167	0,718	33,13
WP-sensor	2,167	0,577	26,647	3,083	0,669	21,68
WP2_50-ipipe	4,000	0,953	23,837	7,083	1,676	23,67
WP2_50-sensor	3,333	0,888	26,629	5,917	1,782	30,11
WP2_100-ipipe	4,250	0,866	20,377	7,500	1,508	20,1
WP2_100-sensor	3,083	1,084	35,145	5,500	1,679	30,52

bajas es Ninguno, a excepción de la escena del paisaje donde su puntuación se encuentra sobre el WP-ipipe por 0.084. Además, para las escenas Pizarra, Muñecos y Retrato en los algoritmos superiores, aquellos que utilizan ganancia del ipipe obtienen las calificaciones más altas con respecto a los que usan las ganancias del sensor.

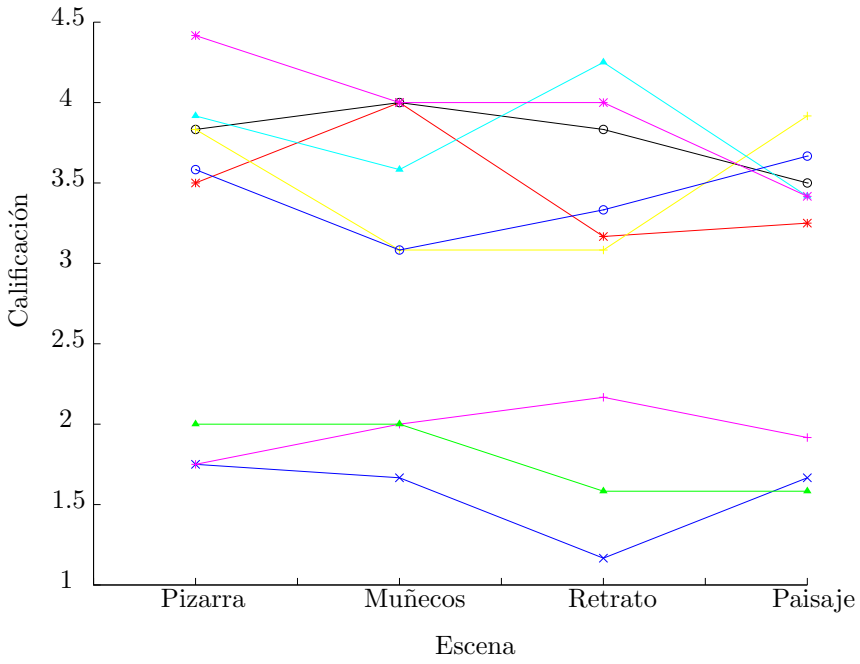


Figura 4.7: Calificaciones de los algoritmos de AWB: Ninguno (x,azul), GW-sensor (*,rojo), GW-ipipe (o,negro), WP-sensor (+,magenta), WP-ipipe (^,verde),WP2_100-sensor (+,amarillo), WP2_100-ipipe (^,celeste), WP2_50-sensor (o,azul), WP2_50-ipipe (*,magenta)

4.3.2 Evaluación de los algoritmos de AE

Los resultados del test cualitativo para los algoritmos de AE para la escena de la pizarra se presentan en la Tabla 4.12. El algoritmo con la mejor calificación es Prom, seguido por Ninguno con una diferencia de 0,25. Sin embargo, según en el resultado de las posiciones estos invierten lugares, siendo la imagen con ningún procesamiento la mejor. Los valores

por defecto de la cámara bajo los cuales se captura la imagen Ninguno, permitieron una iluminación adecuada para la escena en general a pesar de la presencia de secciones saturadas, logrando que la imagen fuera evaluada con calificaciones de *Regular* a *Excelente*. Por su parte, el algoritmo Prom logra disminuir la saturación en la imagen a costa de un oscurecimiento en la misma, lo cual genera dos reacciones extremas en las calificaciones de los observadores: aquellos que lo colocan en la mejor posición con la calificación más alta y los que lo colocan en la posiciones inferiores con calificación *Regular*. Sin embargo, la mayoría de los observadores, ocho de doce, prefieren el oscurecimiento sobre la saturación de la imagen.

El algoritmo Spot fue elegido como el peor para ajustar el tiempo de exposición, obteniendo una calificación de 1,167. En general los algoritmos que basan su ajuste en la presencia de un objeto principal en el centro de la imagen (S, P, Seg), obtienen las calificaciones y posiciones inferiores. En el centro de esta imagen se encuentra ubicado un armario oscuro que da sombra a los objetos en su interior ocasionando que este tipo de algoritmos sobre-expongan la imagen y la saturen para lograr la iluminación adecuada del objeto central.

Tabla 4.12: Calificación de los algoritmos de AE para la escena de la pizarra

Algoritmo	Calificación			Posición		
	Promedio	Desviación Estándar	Error relativo	Promedio	Desviación Estándar	Error relativo
Ninguno	3,917	0,996	25,435	5,000	1,128	22,56
Prom	4,167	1,337	32,091	4,833	1,946	40,27
C-W	3,167	0,835	26,364	4,167	0,577	13,86
P	2,500	1,000	40,000	3,083	0,793	25,72
Seg	2,417	0,900	37,255	2,833	1,030	36,35
S	1,167	0,577	49,487	1,083	0,289	26,65

En la Tabla 4.13 se muestran los resultados para la escena del "chancho". Para esta escena el algoritmo que obtiene la mejor calificación y posición es Seg, el cual es desarrollado precisamente para ajustarse a este tipo de escenas a contraluz, buscando la iluminación adecuada del objeto principal. Sin embargo, la calificación obtenida es de 3,417 ubicándose en la categoría de *Regular* a *Muy Bueno*. A pesar que los objetos principales logran una iluminación adecuada el fondo de la imagen está totalmente saturado, efecto que no es de la preferencia de los observadores. Por el contrario el algoritmo que obtuvo la peor calificación y posición fue Prom, con una calificación de 2,083 (*Regular*). Este algoritmo busca una iluminación promedio en toda la imagen, sin embargo en escenas como la presente se tiene un alto contraste entre el fondo y los objetos principales, así permite ver mayor detalle del fondo a costa del oscurecimiento de los objetos en el primer plano.

Tabla 4.13: Calificación de los algoritmos de AE para la escena del chancho

Algoritmo	Calificación			Posición		
	Promedio	Desviación Estándar	Error relativo	Promedio	Desviación Estándar	Error relativo
Ninguno	2,250	1,357	60,302	2,500	1,508	60,3
Prom	2,083	1,621	77,825	2,167	1,992	91,96
C-W	3,083	1,379	44,723	3,667	1,303	35,53
P	3,333	1,073	32,193	4,333	1,303	30,06
Seg	3,417	1,084	31,716	4,333	1,371	31,63
S	3,000	1,758	58,603	4,000	1,651	41,29

La escena del retrato, como se observa en la Tabla 4.14, obtuvo como mejor algoritmo el

P, con una calificación de 4,17. Además los algoritmos S y C-W, siguientes en el posicionamiento, obtienen calificaciones de 4 y 3,92 respectivamente, con diferencias respecto a P de 0,17 y 0,25, ubicándose aproximadamente en la categoría de *Muy Bueno*. Estos algoritmos toman en cuenta la iluminación en el centro de la imagen, en este caso la cara del retrato y se ajustan para que sea iluminada adecuadamente, efecto deseado para un retrato donde el objeto de interés es la cara de la persona. Por otro lado, la imagen ubicada en la última posición y con la peor calificación es Seg. Esta escena muestra un caso de fallo de este algoritmo, el fondo de la imagen posee una ventana saturada que no afecta la iluminación del objeto principal y la persona en el retrato viste ropa oscura y posee un cabello oscuro, causando que el algoritmo asuma un caso de contraluz donde no lo hay, llevando a una sobre iluminación del rostro de la persona retratada. Asimismo la imagen sin procesamiento obtuvo una calificación de 1,92, con una diferencia de 0,42 con respecto a Seg. Al contrario de Seg en la imagen sin procesar la persona retratada se encuentra oscurecida, sub-expuesta.

Tabla 4.14: Calificación de los algoritmos de AE para la escena del retrato

Algoritmo	Calificación			Posición		
	Promedio	Desviación Estándar	Error relativo	Promedio	Desviación Estándar	Error relativo
Ninguno	1,92	0,79	41,372	2,17	0,94	43,27
Prom	2,17	0,94	43,266	2,5	1	40
C-W	3,92	1,08	27,667	4,75	0,75	15,87
P	4,17	0,72	17,226	5,25	0,75	14,36
Seg	1,5	0,8	53,182	1,58	0,67	42,22
S	4	1,04	26,112	4,75	1,36	28,56

Los resultados de la evaluación cualitativa de los algoritmos de AE para la escena del paisaje se muestran en la Tabla 4.15. Para esta escena la imagen Ninguno está totalmente sobre expuesta, obteniendo un cuadro blanco, ya que la iluminación de esta escena es mayor que las anteriores por lo cual esta escena no es evaluada. El peor de los algoritmos evaluados es S, con una calificación de 2,25 y el mejor es el Prom con una calificación de 4,58. Para este tipo de escena donde no existen objetos principales, los observadores prefieren el algoritmo que toma en cuenta la iluminación de toda la escena y su grado de preferencia va disminuyendo a los algoritmos que toman una región menor de la imagen para calcular la iluminación.

Tabla 4.15: Calificación de los algoritmos de AE para la escena del paisaje

Algoritmo	Calificación			Posición		
	Promedio	Desviación Estándar	Error relativo	Promedio	Desviación Estándar	Error relativo
Ninguno	1	0	0,000	1	0	0
Prom	4,58	0,67	14,587	5,42	1,24	22,89
C-W	3,33	1,07	32,193	4,33	0,78	17,97
P	2,33	0,78	33,364	2,58	0,51	19,93
Seg	3,83	0,72	18,724	5,08	0,29	5,68
S	2,25	0,75	33,501	2,42	0,51	21,31

En la Figura 4.8 se muestran gráficamente las calificaciones obtenidas por algoritmo para cada una de las escenas utilizadas en la evaluación. Seguido del nombre del algoritmo, se indica entre paréntesis el símbolo y el color utilizado para su representación. Al contrario de los algoritmos de AWB, los algoritmos de AE no tienen definidos posicionamientos altos

y bajos. Se evidencia que un mismo algoritmo puede obtener calificaciones superiores a 4 en una escena e inferiores a 1,5 en otra, por tanto muestra la dependencia que tienen los resultados de los algoritmos de AE a la escena.

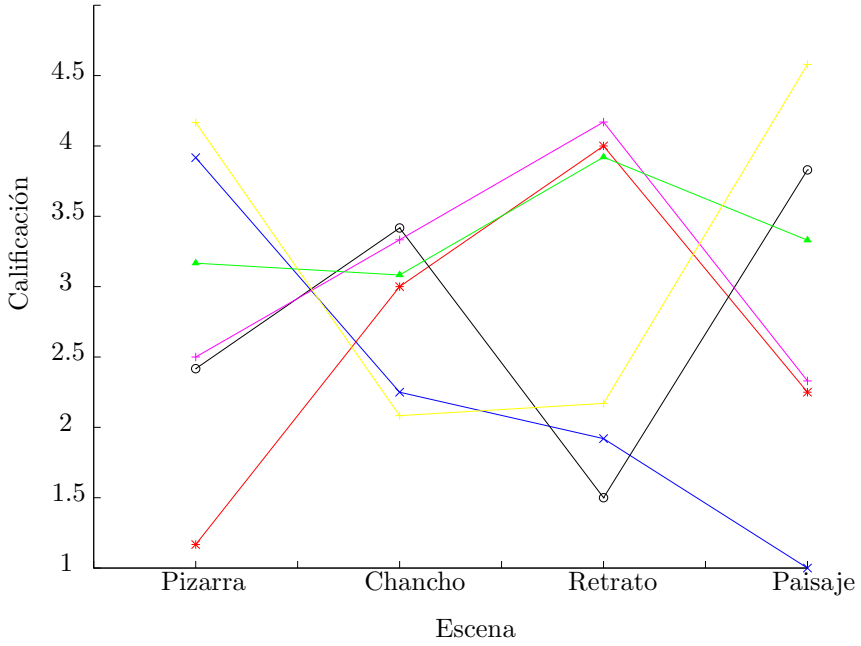


Figura 4.8: Calificaciones de los algoritmos de AE: Ninguno (x,azul), S (*,rojo), Seg (o,negro), P (+,magenta), C-W(ˆ,verde), Avg(+,amarillo)

4.3.3 Uso del CPU

Con el fin de conocer la carga sobre el CPU en la Leopard Board DM365 de la aplicación implementada para el AWB y AE, se obtuvieron los porcentajes de uso del CPU del IPIPE bajo todas las combinaciones posibles de algoritmos. Para realizar la prueba se hizo uso de la herramienta *top*, iterandola diez veces para obtener el promedio de consumo.

Se realizaron tres pruebas de consumo del CPU variando entre ellas el tiempo entre iteraciones de los algoritmos de AEW. Así, en las Tablas 4.16, 4.17 y 4.18 se muestran los resultados para los tiempos de 50, 100 y 500 ms respectivamente. El porcentaje de segmentación para todos los resultados presentados es de 100%, a excepción de WP2_50-ipipe y WP2_50-sensor que utilizan 50%.

De los resultados mostrados se observa que el porcentaje que se obtiene sin ningún algoritmo ejecutandose, es decir, cuando solamente se recolectan los datos estadísticos corresponden a un valor representativo con respecto a los obtenidos durante la ejecución de estos. Por ejemplo para la Tabla 4.18 el porcentaje de Ninguno es del 1,5% mientras que para el resto de los casos el porcentaje promedio es de aproximadamente 2%.

Para todas las tablas se hace evidente que los resultados que utilizan una segmentación del 50% logran un uso del CPU de aproximadamente la mitad de aquellos que usan un 100% de segmentación. La segmentación afecta la cantidad de iteraciones requeridas para obtener

los datos estadíticos de las imágenes, que según los resultados son los que utilizan un mayor porcentaje del CPU.

En la Tabla 4.16 la combinación de algoritmos que consume mayor CPU es el Seg con WP2_100-sensor con 16,3%, estos algoritmos son los que realizan mayor cantidad de operaciones para calcular los ajustes.

Tabla 4.16: Porcentaje de uso del cpu con un tiempo entre iteraciones de 50 ms

AE\AWB	Ninguno	GW-sensor	GW-ipipe	WP-sensor	WP-ipipe	WP2_100-sensor	WP2_100-ipipe	WP2_50-sensor	WP2_50-ipipe
Ninguno	10,3	12,3	12,2	12,8	12,1	14,0	16,0	7,8	6,5
Prom	11,5	13,2	12,8	12,4	12,8	15,4	16,1	7,3	8,7
C-W	11,7	12,1	13,0	11,4	12,7	15,4	16,0	8,8	8,8
P	11,5	12,0	12,1	12,6	12,6	14,9	14,4	7,6	8,5
Seg	12,4	12,4	12,9	12,9	13,0	16,3	14,3	8,9	8,9
S	11,9	12,3	13,2	13,5	11,1	14,6	15,9	7,1	4,4

Para el consumo del CPU con el tiempo entre iteraciones de 100 ms, resultados de la Tabla 4.17, de igual manera los algoritmos con mayor uso del CPU son Seg y WP2_100-sensor, consumiendo 9,1%.

Tabla 4.17: Porcentaje de uso del cpu con un tiempo entre iteraciones de 100 ms

AE\AWB	Ninguno	GW-sensor	GW-ipipe	WP-sensor	WP-ipipe	WP2_100-sensor	WP2_100-ipipe	WP2_50-sensor	WP2_50-ipipe
Ninguno	6,3								
Prom	7,3	8,7	7,8	7,7	7,4	8,9	8,4	5,0	3,6
C-W	6,0	6,5	6,6	7,8	6,4	8,6	7,6	4,8	3,4
P	6,2	7,7	6,9	6,8	6,9	8,7	8,2	4,7	3,3
Seg	6,9	8,6	7,6	7,3	7,4	9,1	8,4	5,1	4,2
S	6,7	8,6	7,8	7,9	7,9	9,0	8,6	4,9	4,0

Igualmente que para los casos anteriores para la Tabla 4.18 Seg con WP2_100-sensor son los algoritmos que realizan el mayor uso del CPU 2,9%.

Tabla 4.18: Porcentaje de uso del cpu con un tiempo entre iteraciones de 500 ms

AE\AWB	Ninguno	GW-sensor	GW-ipipe	WP-sensor	WP-ipipe	WP2_100-sensor	WP2_100-ipipe	WP2_50-sensor	WP2_50-ipipe
Ninguno	1,5	2,1	1,8	2,1	2,0	2,3	2,0	0,9	0,7
Prom	1,8	2,1	2,0	2,2	2,1	2,2	2,7	1,1	1,2
C-W	1,5	2,0	2,0	2,2	2,0	2,2	2,1	1,2	0,7
P	1,8	1,9	2,0	2,2	2,1	2,2	2,0	0,9	0,7
Seg	2,1	2,0	2,0	2,1	1,9	2,9	2,5	0,8	0,8
S	2,2	2,0	2,2	2,3	2,4	2,4	2,2	0,8	0,8

Es claro que conforme se aumenta el tiempo entre las iteraciones el consumo del CPU va a disminuir. Sin embargo, los ajustes tanto de balance de blancos como de la exposición van a tomar mayor tiempo.

Capítulo 5

Conclusiones y Recomendaciones

En el presente capítulo se exponen las principales conclusiones obtenidas al realizar el proyecto y las recomendaciones para futuros trabajos.

5.1 Conclusiones

Se ha presentado una aplicación que ajusta la captura de imágenes en una tarjeta Leopard Board DM365 de acuerdo a las condiciones de iluminación, utilizando para ello las capacidades que ofrece la tarjeta.

Se ha mostrado la biblioteca AEW que implementa tres algoritmos de auto-balance de blancos y un algoritmo de auto-exposición con cinco sistemas de medición de iluminación.

Además, se ha expuesto la implementación una interfaz para configurar y controlar por medio de parámetros el hardware de procesamiento de video de la Leopard DM365 y el del sensor de cámara mt9p031.

La comparación de las imágenes procesadas por los algoritmos y aquellas sin procesar se ha desarrollado por medio de evaluaciones cualitativas y cuantitivas de escenas diversas. En estas evaluaciones se han valorado los cambios en la crominancia y la luminancia de las imágenes.

Los algoritmos de auto-balance de blancos logran mejorar la crominancia del video capturado. La excepción es el algoritmo "Parche Blanco" para escenas reales.

El algoritmo "Parche Blanco" logra mejorar la crominancia de una imagen en escenas controladas, como las utilizadas para la evaluación objetiva, donde se asegure la ausencia de pixeles saturados.

Para el auto-balance de blancos no se puede definir el mejor algoritmo, porque sus resultados son dependientes de la escena. Sin embargo, los resultados de los algoritmos a excepción del "Parche Blanco" logran calificaciones superiores a "Regular" en todas las escenas probadas.

Los resultados de los algoritmos de auto-exposición implementados sobre la luminancia de

un video son dependientes al tipo de escena. Se debe escoger el algoritmo que mejor se ajuste a la escena, porque un algoritmo que obtiene resultados de "Muy Bueno" en una escena puede obtener resultados de "Malo" en otra.

El algorito de auto-exposición con sistema de medición Prom obtiene los mejores resultados para escenas con ausencia de un objeto principal y con bajo contraste.

Para escenas que cuentan con la presencia de un objeto principal los algoritmos S, C-W y P logran obtener la luminancia adecuada.

El algoritmo Seg obtiene el mejor resultado para la escena de alto contraste. Sin embargo, este algoritmo se puede ver engañado por escenas como la del retrato donde el alto contraste en la escena no se debe a un efecto de contraluz generando imágenes con mala luminancia.

El consumo del CPU es dependiente del porcentaje de segmentación de la imagen y del tiempo entre iteraciones.

5.2 Recomendaciones

Aún cuando los algoritmos de auto-exposición logran mejorar la luminancia en las escenas adecuadas para ellos, el promedio general de calificación es 3 lo que corresponde a "Regular". Se debe probar si variando el valor de la luminancia óptima se mejora esta calificación o si es necesaria la implementanción de algoritmos más complejos para mejorarla considerando siempre que se ejecutan en un hardware con limitaciones de procesamiento

Para escenas de alto contraste se deben buscar algoritmos alternativos al Seg, que logren ajustar la luminancia de forma más adecuada.

En general, se pueden implementar algoritmos de mayor complejidad para observar el costo-beneficio entre el mejoramiento en las características de la imagen y la carga en el CPU

Bibliografía

[1] Cubo rgb. Imagen en línea, June 2010. URL http://commons.wikimedia.org/wiki/File:Avl3119color4a.svg.

[2] Aptina Imaging. *MT9P031: 1/2.5-Inch 5Mp CMOS Digital Image Sensor*, 2008.

[3] International Imaging Industry Association. Cpiq initiative phase 1 white paper: Fundamentals and review of considered test methods. *I3A CPIQ Phase 1 document repository*, October 2007.

[4] Kobus Barnard. *Practical Colour Constancy*. PhD thesis, Simon Fraser University, Escuela de Computación,, 1999.

[5] S. Battiato, G. Messina, and A. Castorina. Exposure correction for imaging devices: an overview. In *Single-Sensor Imaging: Methods and Applications for Digital Cameras*, chapter 12. Rastislav Lukac, October 2008.

[6] LLC Black Ice Software. Hsicolormodel. Imagen en línea, 2011. URL http://www.blackice.com/colorspaceHSI.htm.

[7] Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Academic Press, Inc, 2nd edition, 1990.

[8] B Funt and L Shi. The rehabilitation of maxrgb. *Proc. IS&T Eighteenth Color Imaging Conference*, November 2010.

[9] Rafael Gonzalez and Richard Woods. *Digital Image Processing*. Prentice Hall, Inc, 2nd edition, 2002.

[10] Lee J.S., Jung Y.Y, Kim B.S., and Ko S.J. An advanced video camera system with robust af, ae, and awb control. *IEEE Transactions on Consumer Electronics*, 47:694–699, August 2001.

[11] Edmund Y. Lam. Combining gray world and retinex theory for automatic white balance in digital photography. *Consumer Electronics, 2005. (ISCE 2005). Proceedings of the Ninth International Symposium on*, pages 134–139, June 2005.

[12] Edmund Y. Lam and George S. K. Fung. Automatic white balancing in digital photography. In *Single-Sensor Imaging: Methods and Applications for Digital Cameras*. Taylor & Francis Group, LLC, 2009.

[13] M. Mancuso and S. Battiato. An introduction to the digital still camera technology. *ST Journal of System Research*, 2:1–9, 2001.

[14] Lindsay Martin. Test images for computational colour constancy, 2002. URL http://www.cs.sfu.ca/~colour/data/colour_constancy_test_images/index.html.

[15] M Murakami and N. Honda. An exposure control system of video cameras based on fuzzy logic using color information. *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems, 1996*, 3:2181–2187, September 1996.

[16] P Pelgrims, B. Van de Vondel, and G. Van de Velde. Auto exposure, June 2005. URL http://emsys.denayer.wenk.be/emcam/autoexposure.pdf.

[17] H. Pennington, D Wheeler, J. Palmieri, and C. Walters. D-bus tutorial. Tutorial en línea. URL http://dbus.freedesktop.org/doc/dbus-tutorial.html.

[18] Nitin Sampat, Shyam Venkataraman, Thomas Yeh, and Robert L. Kremens. System implications of implementing auto-exposure on consumer digital cameras. *Proc. SPIE. Sensors, Cameras, and Applications for Digital Photography*, 3650:100–107, March 1999.

[19] M. Schimek, B. Dirks, H. Verkuil, and M. Rubli. Video for linux two api specification, 2008. URL http://v4l2spec.bytesex.org/spec/book1.htm.

[20] Wim Taymans, Steve Baker, Andy Wingo, Ronald S. Bultje, and Stefan Kost. *GStreamer Application Development Manual (0.10.32)*.

[21] Texas Instruments. *TMS320DM36x Digital Media System-on-Chip (DMSoC) Video Processing Front End (VPFE) User's Guide*, April 2010.

[22] Quoc Kien Vuong, Se-Hwan Yun, and Suki Kim. A new auto exposure and auto white-balance algorithm to detect high dynamic range conditions using cmos technology. *Proc. of the World Congress on Engineering and Computer Science*, 2008.

[23] Ching-Chih Weng, H. Chen, and Chiou-Shann Fuh. A novel automatic white balance method for digital still cameras. *IEEE International Symposium on Circuits and Systems, 2005. ISCAS 2005.*, 4:3801–3804, July 2005.

Apéndice A

Flujos de datos de GStreamer utilizados

Durante el desarrollo del presente proyecto se utilizaron dos líneas de proceso de gstreamer, las cuales se ejecutaron utilizando gst-launch:

Flujo de captura y despliegue de video

gst-launch -e v4l2src always-copy=false ! video/x-raw-yuv,format= \(fourcc\)UYVY, width=640, height=480, framerate=\(fraction\)30/1 ! TIDmaiVideoSink videoOutput=composite videoStd=D1$_N TSC$

Captura, encodificación y almacenamiento de una imagen png

gst-launch v4l2src always-copy = false ! video/x-raw-yuv,format=\(fourcc\)UYVY, width=640, height=480, framerate=\(fraction\)30/1 ! ffmpegcolorspace ! pngenc ! filesink location = test.png

Apéndice B

Instrucciones de la Evaluación Cualitativa

En la evaluación cualitativa de imágenes se presentaron a los observadores las siguientes instrucciones:

Indicaciones para la evaluación

Seguidamente se le van a presentar varios grupos de imágenes. Cada grupo posee fotografías de una escena específica tomada ante diferentes ajustes de la cámara. Por favor, observe todas las imágenes y evalúelas de acuerdo a las siguientes instrucciones:

1. Cada grupo posee un nombre que lo identifica. Según la primera palabra del grupo se le va a pedir que preste atención a un aspecto distinto de la imagen.

    - Si inicia con WB, preste atención a los colores de la escena, especialmente a la naturalidad de los mismos (la similitud entre la fotografía y la escena real), juzgue si logra reproducir los colores del escenario, sin la presencia de colores dominantes.
    - Si inicia con AE, preste atención a la iluminación de la imagen, juzgue que tan adecuada se encuentra, si no existen pérdidas de detalles por falta o exceso de luz (brillo) especialmente en el objeto principal de la fotografía en caso de poseerlo.

2. Cada una de las imágenes posee en su parte inferior una escala de estrellas para la calificación. La estrella de más a la izquierda corresponde a la menor puntuación (1), mientras la de más a la derecha corresponde a la mayor puntuación (5). Califique conforme a la siguiente escala:

    - 5 - Excelente es claro para usted que la imagen cumple totalmente las características indicadas en el punto 1 según al grupo al que pertenece. Tenga en consideración que ésta debe ser una elección fácil, tiene poca o ninguna duda que la imagen abarca los aspectos indicados.
    - 4 - Muy bueno:
    - 3 - Bueno:

- 2 - Regular :
- 1 - Malo: es claro para usted que la imagen no cumple ninguno de los aspectos indicados en el punto 1, la imagen se ve mal.

3. Tomando en cuenta los aspectos indicados según sea el caso ordene las imágenes de peor a mejor, el orden de calificación corresponde al orden de preferencia de forma ascendente. Es decir la primera imagen a evaluar en la que considere se ve peor.

4. Si desea ver alguna imagen con mayor detalle, puede dar clic sobre ésta para ampliarla.

5. Tome en consideración que una vez calificada la imagen, ésta desaparecerá.

Apéndice C

Relación entre el tiempo de exposición y el ancho del obturador para el sensor mt9p031

Para el sensor mt9v136 se puede calcular el tiempo de exposición(EXP^t) en relación con el ancho del obturador(SW), siguiendo la siguiente ecuación [2]:

$$EXP^t = SW \times ROW^t - SO \times 2 \times PIXCLK^t \tag{C.1}$$

donde $PIXCLK^t$ corresponde a un periodo de reloj de 10.42ns y:

$$\begin{aligned} &SO = 208 \times (Row\_Bin + 1) + 98 + MIN(SD, SDMAX) - 94 \\ &SD = Shutter\_Delay + 1; \\ &SDMAX = 1232, \text{ si } SW < 3; \text{ si no } SDMAX = 1054 \end{aligned} \tag{C.2}$$

Para las condiciones el proyecto se mantienen constantes: $Shutter\_Delay = 0$, $Row\_Bin = 3$, $ROW^t = 37,05\mu s$ y SW siempre es mayor que 3.

Índice alfabético