UNIVERSIDAD NACIONAL DEL ALTIPLANO PUNO

ESCUELA DE POSTGRADO

PROGRAMA DE MAESTRÍA

MAESTRÍA EN INFORMÁTICA

FRAMEWORK MULTIPLATAFORMA PARA PROCESAMIENTO DE IMAGENES Y RECONOCIMIENTO DE PATRONES BASADO EN HTML5

PRESENTADA POR:

RAMIRO PEDRO LAURA MURILLO

PARA OPTAR EL GRADO ACADÉMICO DE:

MAGISTER SCIENTIAE EN INFORMÁTICA

MENCIÓN EN GERENCIA DE TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES

PUNO, PERÚ

2015

UNIVERSIDAD NACIONAL DEL ALTIPLANO PUNO

ESCUELA DE POSTGRADO

PROGRAMA DE MAESTRÍA

MAESTRÍA EN INFORMÁTICA

TESIS

FRAMEWORK MULTIPLATAFORMA PARA PROCESAMIENTO DE IMAGENES Y RECONOCIMIENTO DE PATRONES BASADO EN HTML5

PRESENTADA POR:

RAMIRO PEDRO LAURA MURILLO

PARA OPTAR EL GRADO ACADÉMICO DE:

MASTER SCIENTIAE EN INFORMATICA

MENCIÓN EN GERENCIA DE TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES

APROBADA POR EL SIGUIENTE JURADO:

PRESIDENTE

.....

Dr. Yalmar T. Ponce Atencio

PRIMER MIEMBRO

.....

M.Sc. Nayer Tumi Figueroa

SEGUNDO MIEMBRO

.....

M.Sc. Edgar Holguin Holguin

ASESOR DE TESIS

.....

Dr. Vladimiro Ibañez Quispe

Puno, 29 Enero de 2015

DEDICATORIA

Agradecer a mis amigos de la Universidad, los que siempre han apostado por mí, por cuanto he podido apoyarlos y seguir creciendo juntos. A mis amigos Romel, Sandrita, Linsay, Sra. Dorkita, Yojis, Lichita, Fred, Leo y especialmente a Liseth, ella me ha apoyado bastante en investigación y ahora tengo dos artículos publicados en la Revista de la UNA, a ti mi más sincera gratitud y gracias a todos ustedes por el voto de confianza.

Mi eterna memoria a una mujer muy peculiar que ha puesto mi vida de cabeza, he aprendido lo obscuro al conocerla, ella es de esas personas tóxicamente adictivas que recordaré aunque el tiempo nos separe, gracias Rosely por todos estos años de conocerte y que formes parte de mi vida "La vita spericolata quele giorno 30, quanto tempo e ancora nessuno come te"

Mi eterna gratitud a mis padres Pedro y Adela, soy perfeccionista y obsesivo en el mejor sentido gracias a ellos, recordarle a mi Padre que somos la consecuencia directa de nuestras acciones y si recién ahora somos compañeros en la Maestría y el pronto a sustentar creo que herede lo "amarillo" que tiene pero el estará feliz de saber que voy un paso adelante. Gracias Papá…Recordar siempre a mis preciosas hermanas Marivel y Rocio mis confidentes y con quienes siempre encontraba un momento y rincón para reír y jugar aunque el mundo reniegue, con gatos o con perros compartiendo la vida, desearía que no se vuelvan tan renegonas como la adultez vuelve a las personas, sonrían conmigo que poco humor tiene este grisáceo mundo, saben cuánto las quiero a pesar de que ahora nos vemos poco.

Ramiro Pedro

AGRADECIMIENTOS

- A mis Docentes de la Maestría en Informática de la UNA Puno, en especial al Ing. Nayer Tumi Figueroa, gracias por su apoyo y sobre todo los libros que ha compartido con mi persona, han contribuido para mejorar mi Inglés y los papers y consejos que nos ha sabido compartir, agradecerle también por la gran cantidad de experiencias en congresos y eventos en lo que siempre hemos trabajado hombro a hombro.

- A mis colegas de trabajo en el JAE Salcedo, al *M.Sc. Jorge Condori* y el *M.Sc. Leonid Aleman Gonzales* por tenderme la mano y su confianza. Así como a mis colegas en la Universidad Privada San Carlos al Lic. Yerko Arpasi por permitirme colaborar con la carrera mediante la investigación.

- A mis jurados de tesis al *D.Sc. Yalmar Ponce Atencio, M.Sc. Nayer Tumi Figueroa, M.Sc. Edgar Holguin Holguin* y mi asesor de tesis el experimentado investigador *Dr. Vladimiro Ibañez Quispe.* Sus correcciones han permitido que finalmente se sustente y valide esta investigación.

- Al Directo de la Maestría en Informática al M.Sc. Alejandro Apaza Tarqui, agradecerle la agilización de los trámites propios de la sustentación y posterior obtención del grado, mencionarle que aunque hemos tenido nuestras diferencias de ideales, somos profesionales que buscan un bien común el renombre de nuestra carrera y ahora como Maestros Investigadores.

ÍNDICE GENERAL

ÍNDICE DE CUADROS

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

RESUMEN

En esta investigación presentamos el análisis y desarrollo de un Framework para el procesamiento de Imágenes y Patrones basado en la potencia del lenguaje HTML5 y las capacidades orientada a objetos del lenguaje Javascript, también la importancia de su funcionalidad sobre cualquier sistema operativo, mejorando y simplificando así el uso de técnicas de procesamiento de imágenes con el uso del Framework mediante la herencia o instanciación simple, el acceso directo a las funciones miembro y propiedades de la imagen. La implementación orientada a objetos simplificó el acceso a las variables y características de las imágenes (anchura, altura, profundidad de color y mapa de bits), Se ha adaptado los algoritmos de cálculo y procesamiento de matrices para realizar procesos de convolución y discretización de color, así también el uso de filtros Gaussianos y Operador de Sobel, Canny, Roberts. Dejando preparados los datos para localizar patrones y obtención de histogramas de color. Se ha obtenido como resultado una implementación clara, compacta. Para poder derivar y ejecutar sobre cualquier navegador que soporte la tecnología HTML5 y sobre todo cualquier sistema operativo. Sobre todo respecto a otros Framework CGI basados en C++ y Phyton ofrecemos mayor flexibilidad y portabilidad.

Palabras claves:, CGI, Framework,HTML5, Procesamiento de Imágenes, WebApp.

ABSTRACT

This paper presents the analysis and development of a Framework for Image Processing and Patterns, based on the power of language HTML5 and JavaScript language, also the importance of its functionality on any operating system oriented capabilities, improving and simplifying the use of image processing techniques with the use of this Framework through inheritance or simple instantiation, direct access to the member functions and all images properties. Also the object-oriented implementation will simplify access to the variables and images characteristics (width, height, color depth and bitmap), computational algorithms and processing matrices are adapted to perform convolution processes and color discretization, so the use of Gaussian filter and Sobel operator, Canny, Robert leaving ready for locating data patterns and color histograms obtained. Was obtained as a result a clear, compact and ready for implementation derived and implemented on any browser that supports HTML5 technology and especially on any operating system. With regard to other Framework-based CGI C++ and Python offer more flexibility and portability.

keywords:CGI , Framework, HTML5, Image Processing, WebApp.

INTRODUCCIÓN

Un Framework es una capa intermedia de desarrollo, soluciona problemas de implementación de procesos primarios para así **simplificar** el objetivo principal de un proyecto de programación. En particular sobre proyectos de procesamiento de imágenes y patrones sobre entornos diferentes plataformas operativas. La implementación basada en WebApp y HTML5 nos permitirá la ejecución sobre los diferentes sistemas operativos y dispositivos como Smartphones, Tablets, Laptops y Desktop PC, pues el único software que requiere es un browser (navegador) que soporte el estándar HTML5. La implementación del proyecto final puede realizarse en PHP, HTML, JSP o cualquier otro lenguaje del lado del servidor o el uso de DOM Ajax de ser necesario, al finalizar y publicar este se ejecutará sin muchos problemas más que de adaptación a las dimensiones del dispositivo.

Para el procesamiento de imágenes se tiene como entrada elementos como <IMG>,<CANVAS>,<VIDEO>y <WebCam>. La colección de funciones de exportación o colección de APIs (Application Program Interface) están implementados en el lenguaje de procesamiento Javascript estos pueden escalarse en la implementación base, se ha documentado los procesos para agregar más funciones. Esto permitirá al programador de acuerdo a su habilidad incrementar la potencia del Framework, pero sobre todo obtener una aplicación portable y ejecutable desde cualquier dispositivo.

CAPÍTULO I

PROBLEMÁTICA DE INVESTIGACIÓN

1.1 PLANTEAMIENTO DEL PROBLEMA

Los Frameworks son colecciones de APIs que permiten resolver un problema de diseño, en este caso en particular se requiere de una colección de APIs para procesamiento de imágenes y reconocimiento de patrones que sea portable a cualquier sistema operativo y observando la tendencia de desarrollo de aplicaciones hacia dispositivos móviles y aplicaciones basadas en Web. Nos planteamos el desarrollo del Framework para entornos Web basándonos en la potencia ofrecida por HTML5, que nos permitirá reducir la complejidad del desarrollo del proyecto y así obtener resultados óptimos.

El procesamiento digital de imágenes es una línea de investigación en la computación, comparado con el estudio de compiladores y sistemas operativos que iniciaron en la década de los años 1950, lo que hace que los conceptos y teoremas para la resolución de nuevos problemas aún estén en desarrollo, debido a que antes de pensar en ello, se tenía que

desarrollar el hardware y los sistemas operativos gráficos que permitieran realizarlo. Por otro lado, los algoritmos y las técnicas de optimización que han tenido que desarrollarse para el procesamiento digital de imágenes tienen que ser sofisticados.

Una imagen contiene elementos numéricos matriciales definidos por el ancho y la altura los cuales permiten que nuestros sentidos visuales puedan interpretarlo como una representación pictórica de la realidad. Este elemento matricial es el color, el cual está representado generalmente por el modelo RGB (componentes Rojo, Verde y Azul) que tiene una representación tridimensional en el espacio del color, y como tal es posible aplicar los conceptos de geometría del espacio y en nuestro caso métodos estadísticos de regresión, medias y series de tiempo para optimizar la transición del color y generar la mejor representación gráfica de ellos, se puede generar un histograma y suavizarlo mediante técnicas estadísticas.

1.2 JUSTIFICACIÓN

Existen Frameworks especializados para acelerar el procesamiento de imágenes, visión computacional y computación gráfica como OpenCV, OpenCL, CUDA, entre otros. Que si bien son ampliamente eficientes y escalables sobre su implementación en el lenguaje C++ orientado a objetos, esto los hace limitados en su ejecución pues requiere un proceso de adaptación de librería al sistema operativo, compilación regida por la

interfaz gráfica host, versión de sistema operativo, arquitectura del procesador y aplicación objetivo.

Por otro lado, al implementar nuestro Framework en los lenguajes HTML5 y Javascript permitiremos la portabilidad, sin tener que realizar cambios, pues al estar trabajando sobre el browser (navegador) este se encarga de la portabilidad de nuestro Framework, por lo que reducimos así la complejidad de adaptación, compilación y así concentrar esfuerzos en la aplicación final soportada por un Framework escalable, portable y sobre todo funcional.

Para el desarrollo, se cuenta la metodología de Programación Extrema para un proceso iterativo durante el análisis, implementación y pruebas de ejecución, mejorando en cada iteración los elementos nuevos para el procesamiento y reconocimiento de patrones.

Esto nos conduce a la pregunta de investigación: **¿El desarrollo de un Framework Multiplataforma aportaráa la solución de portabilidad y accesibilidad a las tareas del Procesamiento de Imágenes y Reconocimiento de Patrones?**

1.3 OBJETIVOS

1.3.1 OBJETIVO GENERAL

Desarrollar un Framework Web multiplataforma basado en la tecnología HTML5 para el procesamiento de imágenes y reconocimiento de patrones.

1.3.2 OBJETIVOS ESPECÍFICOS

- Implementar un Framework Web para CGI que permita desarrollar aplicaciones deprocesamiento de imágenes y/o videos, con funciones APIs sencillas y claras de usar.
- Adaptar las rutinas de procesamiento de imágenes escritas en lenguaje C++, hacia el lenguaje Javascript con interfaz HTML5 portando su funcionalidad.
- Publicar los resultados y la documentación del Framework en un sitio web para las pruebas Online y descargas.

1.4 HIPÓTESIS

El desarrollo de un Framework Web Multiplataforma mejorafavorablemente el desarrollo de software web de Procesamiento Digital de Imágenes y Reconocimiento de Patrones.

CAPÍTULO II

MARCO TEÓRICO

2.1 ANTECEDENTES

Asqui V.,Richard (2009) en su tesis:***Sistema de Reconocimiento de Firmas y Rubricas Digitales***, plantea el uso de filtros basados en la convolución por matrices para la obtención de características principales en las firmas, concluye que el uso de este filtro sumado a las distancias euclidianas nos permite obtener excelentes resultados en la comparación y comprobación de firmas, una vez escaneadas.

Bahamón C., Nelson (2011). En su tesis:***Restauración de Imágenes Mediante un Modelo Matemático Basado en las Técnicas de Detección de Bordes y Propagación de Texturas***. Se utilizó en primera instancia la técnica de síntesis de texturas basada en parches. Se observó que para lograr la mejor restauración posible se debe lograr un equilibrio entre la propagación del color (textura) y de bordes (estructura) ya que el dar prioridad a una u otra genera resultados muy distintos. En otras palabras, estos dos factores “compiten” entre sí. Así mismo se observó mediante experimentos sencillos que el resultado de la

restauración en la imagen depende fuertemente del orden en que se restauren los pixeles de la zona objetivo. Se planteó una función de probabilidad que soluciona ambos problemas, genera un equilibrio entre la propagación de textura y de estructura y determina el orden de restauración de la zona objetivo.

Sardi L., Daniel (2012). En su tesis: ***Algoritmo de Segmentación Online de Imágenes en Secuencias de Video. Universidad de Buenos Aires, Facultad de Ingeniería, Mayo 2012.***Las técnicas de sustracción de fondo son una técnica particular de segmentación deimágenes. Tienen aplicaciones prácticas muy diversas, tales como seguimiento (tracking) de personas y vehículos a través de distintas cámaras, segmentación de lasdistintas escenas de video y compresión de video, entre otras.El propósito de este trabajo es presentar una algoritmo de sustracción de fondo alcanzando mejores tiempos de procesamiento.

2.2 MARCO TEÓRICO

2.2.1 EL COLOR

El color (chroma, chrómatos) es una percepción visual que se genera en el cerebro humano y de animales al interpretar las señales nerviosas que le envían los foto-receptores en la retina del ojo, que a su vez interpretan y distinguen las distintas longitudes de onda que captan de la parte visible del espectro electromagnético (la luz).***(Sardi L. D.,2012)***

Todo cuerpo iluminado absorbe una parte de las ondas electromagnéticas y refleja las restantes. Las ondas reflejadas son captadas por el ojo e interpretadas en el cerebro como distintos colores según las longitudes de ondas correspondientes.El color es un factor importante a la hora de obtener información sobre una escena Para el sistema de visión humano, el color es una sensación. Depende de las propiedades físicas de la luz y un complejo procesamiento del aparato visual. Respecto a los receptores de la luz, tanto cámaras como el propio aparato visual humano, responden sólo a ciertos rangos dentro del espectro electromagnético. El ojo humano únicamente es capaz de percibir el espectro que abarca de los 400 a 750 nanómetros de período.***(Sardi L. D.,2012)***

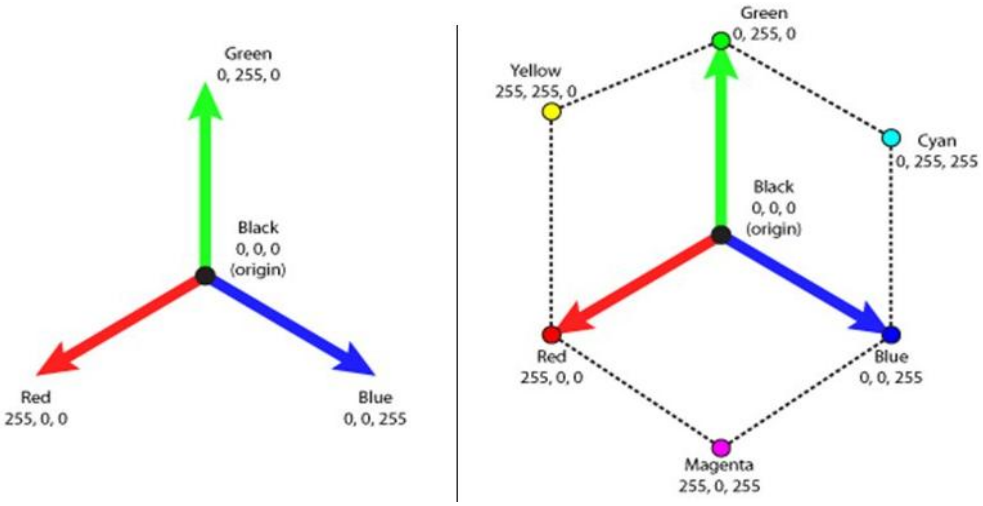
FIGURA 1

Intensidad de luz requerida para la recepción visual determine un color

Color	Longitud de onda
violeta	~ 380-450 nm
azul	~ 450-495 nm
verde	~ 495-570 nm
amarillo	~ 570–590 nm
naranja	~ 590–620 nm
rojo	~ 620–750 nm

FIGURA 2

Modelo de color RGB y CMYK usado para la representación del color en los computadores por CIE-1931 (International Commission on Illumination)



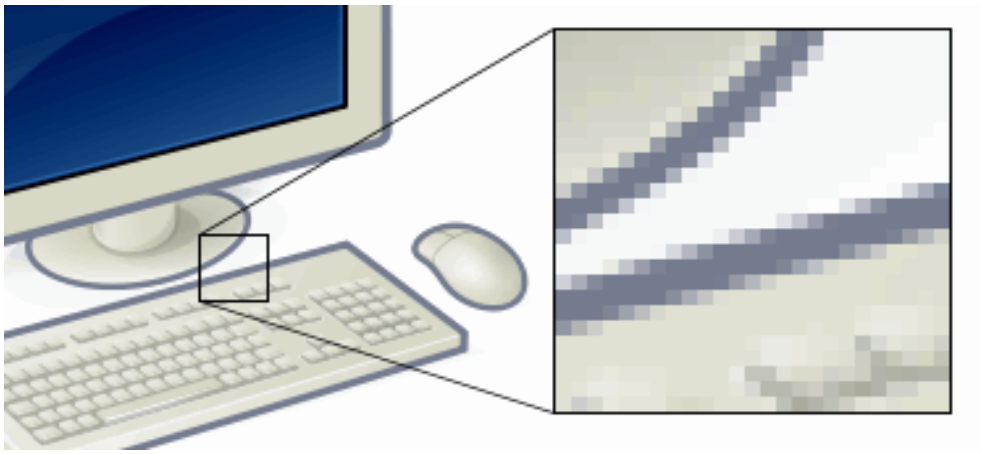
2.2.2 IMAGEN

Es la colección de información pictórica, usualmente está representado como una superficie bidimensional, puede tomarse como un vector en el espacio 2D y hasta 3D en una malla de polígonos que trazan la superficie de un rostro, en el caso de una fotografía se debe tomar como una imagen de mapa de bits (BMP) es decir una colección matricial de puntos, el cual recibe el nombre de Pixel. Podemos tener una descripción visual y gráfica para ello puede observarseen la **Figura 1**.

El término imagen se refiere a una función bidimensional de la luz y la intensidad, a la que indicamos por ***f(x,y)*** da la intensidad (iluminación) de la imagen en este punto puesto que la luz es una forma de energía, ***f(x,y)*** debe ser estrictamente mayor que cero y finita es decir, cumplirá siempre la condición: ***0 < f(x,y) < 256***,

soportando un total de 256 tonalidades de color por elemento. ***(Gonzales et al, 1999)***

FIGURA 3
Ampliación de una zona de una imagen donde se pueden apreciar los pixeles, observe la reunión de puntos que la representan.



2.2.3 DESARROLLO DE FRAMEWORK

Los Framework contribuyen a la simplificación de la interfaz de desarrollo sobre el problema real de procesamiento de imágenes y patrones, como el caso particular de un Framework para sombras y entornos para OpenGL y GLSL (Librería Grafica Libre), dejando al desarrollador concentrarse en el algoritmo y el objetivo planteado. ***(Andrade V. et al, 2013)***

Un Framework para aplicaciones web está diseñado para apoyar el desarrollo de sitios web dinámicos, aplicaciones web y servicios web. Este tipo de Frameworks intenta aliviar el exceso de carga asociado con actividades comunes usadas en desarrollos web. Por ejemplo, muchos de ellos proporcionan bibliotecas para

acceder a bases de datos, estructuras para plantillas y gestión de sesiones, y con frecuencia facilitan la reutilización de código.***(Andrade V. et al, 2013)***

La palabra inglesa "framework" (marco de trabajo) define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.En el desarrollo de software, un framework o infraestructura digital, es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio, y provee una estructura y una especial metodología de trabajo, la cual extiende o utiliza las aplicaciones del dominio. ***(Gauchat J., 2012)***

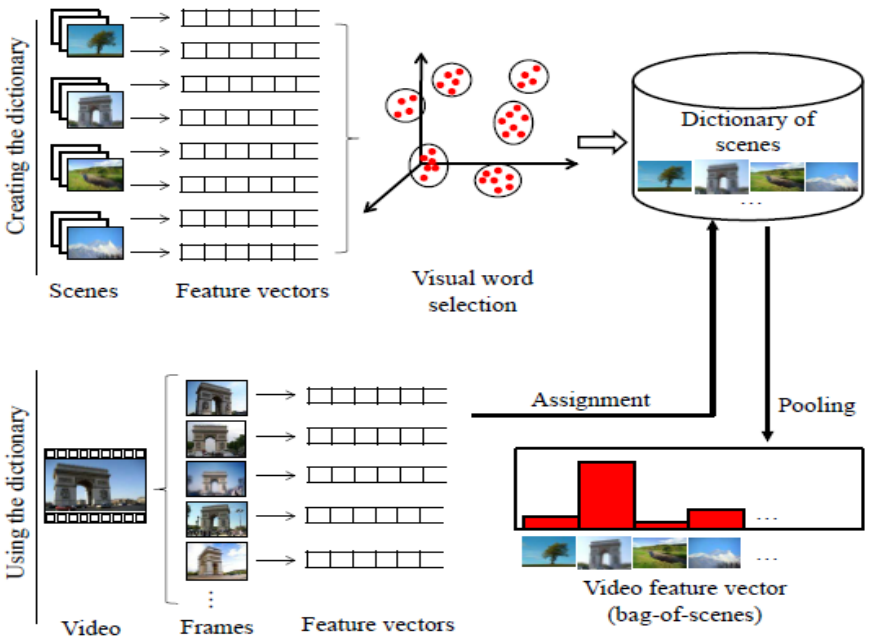
No es más que una base de programación que atiende a sus descendientes (manejado de una forma estructural y/o en cascada), posibilitando cualquier respuesta ante las necesidades de sus miembros, o en secciones de una aplicación (web), satisfaciendo así las necesidades más comunes del programador.

Actualmente los Framework han variado su arquitectura para el caso de los MVC (Modelo Vista Controlador) ya que debemos fragmentar nuestra programación. Tenemos que contemplar estos aspectos básicos en cuanto a la implementación de nuestro sistema. ***(Gauchat J., 2012)***

2.2.4 CARACTERIZACIÓN Y REPRESENTACIÓN DE IMÁGENES

La representación basada en imágenes por contenido ha sido de las investigaciones más retadoras por décadas, con esta tesis contribuimos en la mejora del procesamiento y obtención de características y representación de imágenes. Podemos notar por ejemplo la popularidad de los algoritmos de reconocimiento de rostros embebidos en cámaras digitales y la tendencia de aplicaciones móviles, es aquí donde deberíamos dirigir algunas investigaciones por su tendencia. ***(Otavio A. et al, 2013)***

FIGURA 4
Representación y caracterización de imágenes



2.2.5 DISCRETIZACIÓN DEL COLOR

Se Representaran y almacenaran las imágenes sobre matrices tridimensionales de valores que contendrán tonos de color, deberá tenerse en cuenta que una imagen puede ser binaria, monocromática o multicolor. ***(Gonzales et al, 1998)***

Una imagen binaria es aquella donde cada elemento de la matriz puede tomar el valor de 0 o 1. Entonces, formalmente se define como:

$$B(r,c):[0,R-1]\times[0,C-1]\subset Z^2 \rightarrow [0,1]\subset Z$$

Una imagen monocromática es aquella donde cada elemento toma un valor entero entre 0 y 255, conocido como escala de grises de 256 tonos (grayscale). Formalmente se define como:

$$G(r,c):[0,R-1]\times[0,C-1]\subset Z^2 \rightarrow [0,255]\subset Z$$

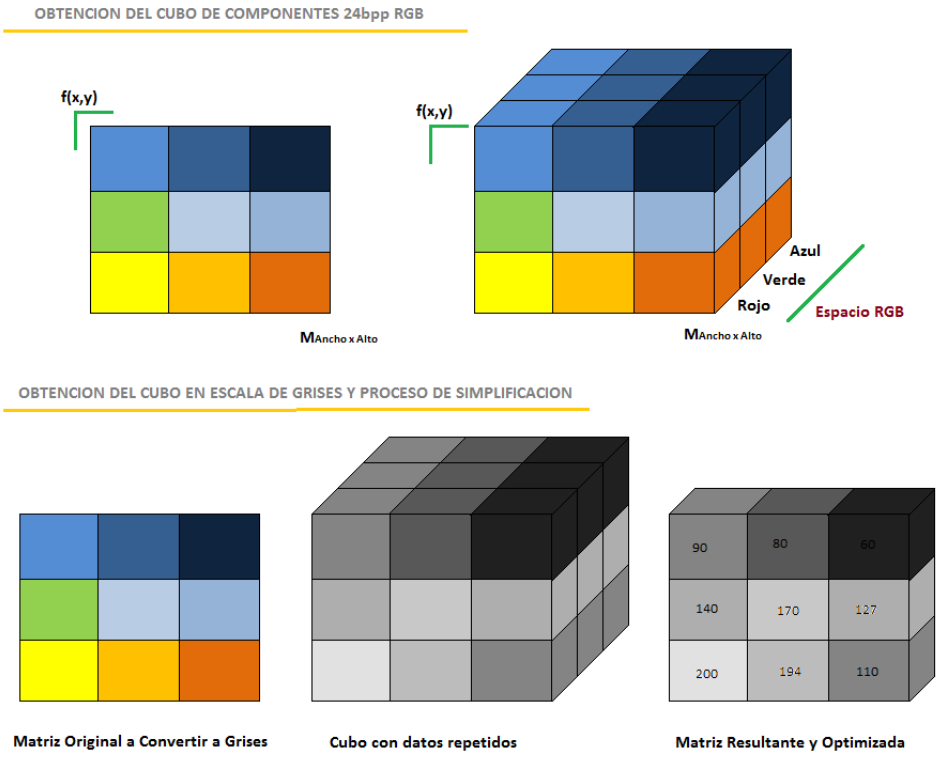
Una imagen color es aquella donde cada elemento toma tres coordenadas RGB (red, green, blue) que compuestos forman un color de un espacio mínimo de 16’777,216+. Formalmente se define como:

$$I(r,c):[0,R-1]\times[0,C-1]\subset Z^2 \rightarrow$$
$$R[0,255]\times G[0,255]\times B[0,255]\subset Z^3$$

Visto gráficamente las matrices multidimensionales de imágenes se representan:

FIGURA 5

Vista Tridimensional del cubo de componentes RGB de una imagen y representación de la obtención de imágenes en escala de grises.



2.2.6 ESPACIO DE COLOR RGB

El espacio RGB (red-green-blue) es una manera de codificar colores en tres bytes, loque da una variedad de 16 millones de códigos diferentes. Se utiliza un byte pararepresentar el componente rojo, otro para el verde y otro para el azul.

Los procesos de sustracción de fondo se basan por lo general en la diferencia de uncuadro de la escena contra el modelo del fondo de la escena.Cuando se trabaja con imágenes monocromáticas,

cada píxel está compuesto de unvalor escalar, y sólo existe un valor para comparar.En las imágenes a color, cada pixel tiene varios atributos, donde cada uno aportainformación distinta.***(Sardi L. D., 2012)***

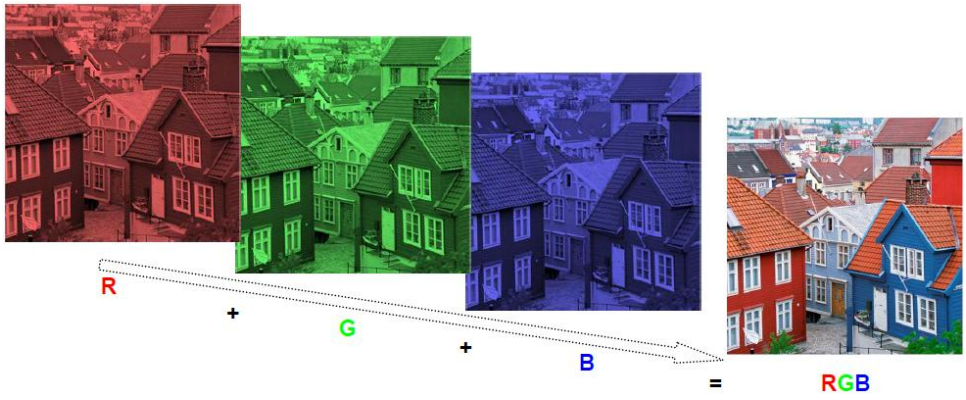
Los pixeles contienen la información del color cada pixel contiene la información de 3 componentes, podemos apreciar en la Figura 6, la distribución matricial de una imagen, si separamos los 3 componentes tendremos 3 imágenes como en la Figura 7, al sumarse generan la imagen RGB compuesta.***(Sardi L. D., 2012)***

FIGURA 6

Estructura de una imagen digital

FIGURA 7

Superposición de las componentes R,G y B en la composición final



Cuando los valores de la profundidad de colores aumentan, se torna impráctico mantener una tabla o mapa de colores debido a la progresión exponencial de la cantidad de valores que el pixel puede tomar. En esos casos se prefiere codificar dentro de cada pixel los tres valores de intensidad luminosa que definen un color cualquiera en el modelo de color RGB.

Para la profundidad de color de 24 bits por pixel, se habla de color verdadero debido a que la policromía se acerca a lo que el ojo humano puede encontrar en el mundo real, y a que dicho ojo humano se torna incapaz de diferenciar entre un tono y otro, si la diferencia se mantiene en un cierto rango mínimo. En la profundidad de color de 24 bits por pixel, se dedica un octeto entero a representar la intensidad luminosa de cada uno de los tres tonos primarios de rojo, verde y azul, lo cual permite que cada pixel pueda tomar ***256x256x256 = 16.777.216*** colores distintos.

Cuando se utilizan 32 bits para representar un color se agrega al esquema de 256 valores para cada tono primario un cuarto canal denominado alfa que representa la transparencia. Este valor se utiliza cuando se deben superponer dos imágenes.Dados los colores:

$$A[r_1, g_1, b_1, a] \; over \; B[r_2, g_2, b_2, a]$$
$$A \; over \; B = [r_1 \times a + r_2 \times (1-a), \; g_1 \times a + g_2 \times (1-a), \; b_1 \times a + b_2 \times (1-a) \;]$$

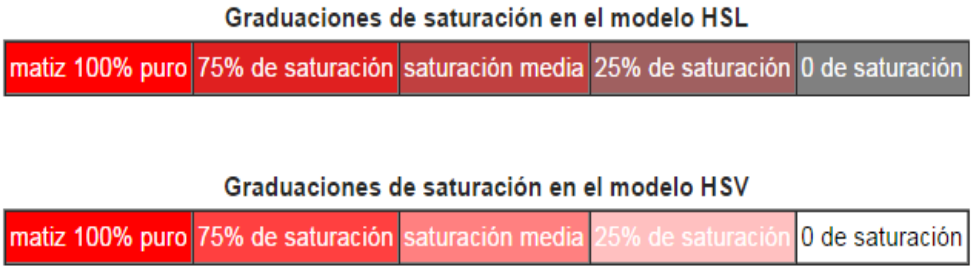
2.2.7 ESPACIO DE COLOR HSL y HSV

HSL (Hue, Saturation, Lightness) es similar al modelo HSV (Hue, Saturation, Value) pero refleja mejor la noción intuitiva de la saturación y la luminancia como dos parámetros independientes, y por tanto es un modelo más adecuado para los artistas. ***(Valdivel A. et al, 2012)***. La especificación de las hojas de estilo en cascada (CSS) en su versión 3 prefieren HSL porque es simétrico al eje luz-oscuridad, lo que no sucede con el modelo HSV ("Advantages of HSL are that it is symmetrical to lightness and darkness (which is not the case with HSV for example)"). Significa que:

En HSL, la componente de la saturación va desde el completamente saturado hasta el gris equivalente, mientras que en HSV, con V al máximo, va desde el color saturado hasta el blanco, lo que no es muy intuitivo.***(Zhong Liu et al, 2012)***

En las aplicaciones de tratamiento de color, los modelos HSV y HSL se representan como unárea lineal o circular para el matiz y la saturación; y un área bidimensional, como un cuadrado o triángulo, para el valor/luminancia. En esta representación las diferencias entre HSV y HSL son irrelevantes. Sin embargo, algunas aplicaciones permiten escoger el color por medio de deslizadores lineales o entradas numéricas. En estos casos solo se usa uno de los dos modelos, comúnmente HSV, dependiendo de la intensidad de luz requerida. ***(Zhong Liu et al, 2012)***

FIGURA 8
Graduación de tonalidades entre los modelos de color HSL y HSV



2.2.8 SEGMENTACION DE IMÁGENES

La segmentación en el campo de la visión artificial es el proceso de dividir una imagen digital en varias partes (grupos de píxeles) u objetos. El objetivo de la segmentación es simplificar y/o cambiar la representación de una imagen en otra más significativa y más fácil de analizar. La segmentación se usa tanto para localizar objetos como para encontrar los límites de estos dentro de una imagen. Más precisamente, la segmentación de la imagen es el proceso de asignación de una etiqueta a cada píxel de la imagen de forma que los píxeles que compartan la misma etiqueta también tendrán ciertas características extraíbles visuales similares.**(Suarez B. A., 2009)**

El resultado de la segmentación de una imagen es un conjunto de segmentos que cubren en conjunto a toda la imagen, o un conjunto de las curvas de nivel extraídas de la imagen (véase la detección de bordes). Cada uno de los píxeles de una región son similares en alguna característica, como el color, la intensidad o la textura. Regiones adyacentes son significativamente diferentes con respecto a la(s) misma(s) característica(s).

Para la segmentación se estudiarán técnicas de segmentación por umbralización y segmentación por detección de regiones. Para la reconstrucción del escenario se estudiarán distintos algoritmos para resolver el problema de la correspondencia. La parte de triangulación basada en fundamentos geométricos, necesaria para el cálculo de las coordenadas de un punto en el espacio a partir de la proyección de ese punto en las dos imágenes, no se aborda en este documento. **(Suarez B. A., 2009)**

FIGURA 9
Imagen original superpuesta con el resultado de la misma imagen usando un segmentado de color HSV

RGB Bitmap	HSV Result

2.2.9 DETECCIÓN DE BORDES

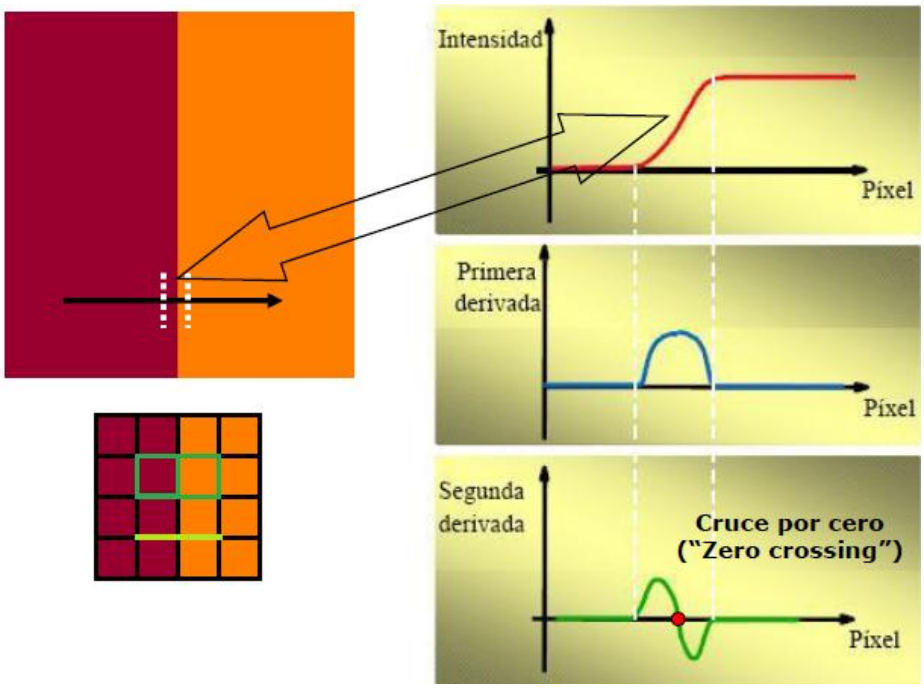
El proceso de detección de bordes sirve para simplificar el análisis de imágenes y drásticamente reducir la cantidad de datos procesador, al mismo tiempo se preserva la información estructura sobre las características del objeto ***(Canny J.,1986)***Un cambio brusco en la intensidad, dará lugar a un máximo o un mínimo en la primera derivada de la función de intensidad o brillo B(x,y) o equivalentemente un cero en la segunda derivada. Estos ceros, son importantes en muchos análisis por ejemplo para la detección de bordes. Un algoritmo conocido para detección de bordes basado en cruces por cero es el algoritmo de Canny. Los siguientes son algunos de los algoritmos operadores mas conocidos para la detección de bordes:

- Operadores Basados en Gradiente
  - Operador de Roberts
  - Operador de Sobel
  - Operador de Prewit
  - Operador Isotropico (Frei Chen)
- Operadores Basados en Cruce por Cero
  - Operador de Marr – Hildreth
  - Operador de Canny

Basados en Zero Crossing:En el análisis de una imagen es importante distinguir sobre la misma, los objetos que se desean estudiar, del resto de la imagen. Las técnicas que se utilizan para

hacer dicha distinción se conocen como técnicas de segmentación. Una de las técnicas de segmentación típicas es la detección de bordes.Una buena detección de bordes facilita la identificación de las fronteras de los objetos de la imagen y por lo tanto su reconocimiento.Al referirse a bordes se puede pensar que lo que se está buscando son contornos, lo cual no es necesariamente cierto. En si el propósito de la detección de bordes es obtener como resultado una imagen donde se resalten los pixeles de aquellos puntos de la imagen original en donde se presentan cambios bruscos de intensidad. ***(Bahamón C. N., 2011)***

FIGURA 10
Representación grafica y de pixeles del cambio de tonalidad, comportamiento de la primera y segunda derivada en un borde. (case operadores Zero Crossing)

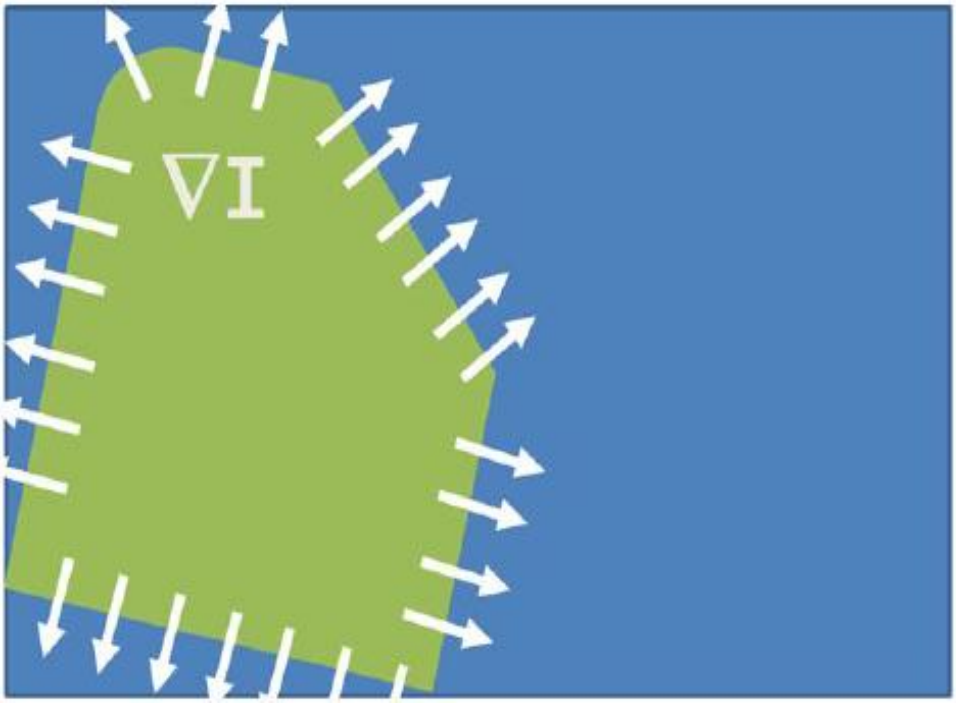


Basados en Gradiente:La magnitud del gradiente en cada punto (pixel) indica que tan marcado está el borde. Así, se puede decir

que un pixel de borde se describe mediante dos elementos (intensidad del borde y dirección del borde). A la imagen obtenida en el espacio de frecuencias se le aplica un filtro pasa alto. Este filtro lo que hace es atenuar las frecuencias bajas y mantener las altas. Puesto que las frecuencias altas corresponden a cambios bruscos de intensidad lo que se está haciendo es realzar los bordes, pues estos contienen gran cantidad de dichas frecuencias. Este filtro también realza los contrastes en la imagen, casi como una curva de nivel. ***(Bahamón C. N., 2011)***

FIGURA 11

Gradiente de una imagen simple con bordes bien definidos. Los vectores indican la dirección en la cual es mayor la razón de cambio en la intensidad de la imagen. Por lo tanto es claro que en las zonas de un solo color el gradiente es cero.



2.2.10 FILTROS POR CONVOLUCIÓN

Convolución es un operador matemático que transforma dos funciones ***f*** y ***g*** en una tercera función. Representa la magnitud en la que se superponen ***f*** y una versión trasladada e invertida de ***g***. Una convolución es un tipo muy general de media móvil.***(Gonzales R., 2008)***. Se define de la forma:

$$F(x,y) = f(x,y) * g(x,y) = \sum_i \sum_j f(x+i, y+j) h(i,j)$$

$$\begin{bmatrix} y[0] \\ y[1] \\ y[2] \\ . \\ . \\ . \\ y[2*n-1] \end{bmatrix} = \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ . \\ . \\ . \\ x[n] \end{bmatrix}^T \begin{bmatrix} h[0] & h[1] & h[2] & ... & h[n] & 0_k & ... & 0_{2*n-1} & & \\ 0 & h[0] & h[1] & h[2] & ... & h[n] & 0_k & ... & 0_{2*n-2} & \\ 0 & 0 & h[0] & h[1] & h[2] & ... & h[n] & 0_k & ... & 0_{2*n-3} \\ . & & & & & & & & & \\ . & & & & & & & & & \\ . & & & & & & & & & \\ 0 & 0 & ... & h[0] & h[1] & h[2] & ... & h[n] & & \end{bmatrix}$$

¿Qué es una matriz de convolución?[1] Es posible hacerse una idea sin usar las herramientas matemáticas que solo conocen unos pocos. Convolución es el tratamiento de una matriz por otra que se llama ***"kernel"***.El filtro matriz de convolución usa una primera matriz que es la imagen que será tratada. La imagen es una colección bidimensional de píxeles en coordenada rectangular. El kernel usado depende del efecto deseado.

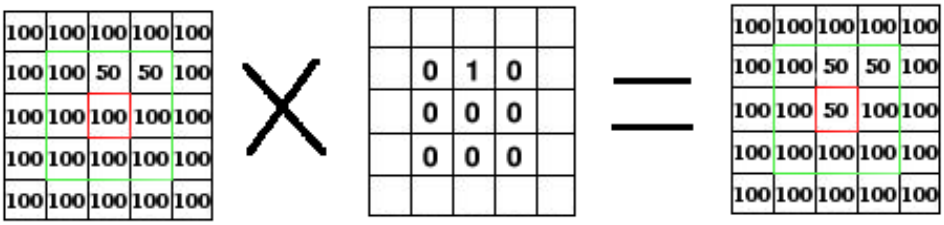
El Software GIMP[2] usa matrices 5x5 o 3x3. Consideraremos solo las matrices 3x3, son las más usadas y son suficiente para los efectos deseados. Si todos los valores de un kernel se

[1] Matriz de Convolución: extraído de *http://docs.gimp.org/2.4/es/plug-in-convmatrix.html*

[2] GIMP: GNU Image Manipulation. Manual de usuario en Español: *http://docs.gimp.org/2.4/es/*

seleccionan a cero, el sistema la considerará como una matriz 3x3.El filtro examina, sucesivamente, cada píxel de la imagen. Para cada uno de ellos, que llamaremos “píxeles iniciales”, se multiplica el valor de este píxel y el valor de los 8 circundantes por el valor correspondiente del kernel. Entonces se añade el resultado, y el píxel inicial se regula en este valor resultante final. Un ejemplo para detallar:

FIGURA 12
Descripción grafica de convolución matricial y obtención del nuevo valor a ser reemplazado sobre el color, una vez procesado.



Lo que sucede: el filtro lee sucesivamente, de izquierda a derecha y de arriba a abajo, todos los píxeles del área de acción del kernel. Se multiplica el valor de cada uno de ellos por el valor correspondiente del kernel y se suman los resultados: (100*0)+(50*1)+(50*0)+(100*0)

+(100*0)+(100*0)+(100*0)+(100*0)+(100*0)+(100*0) = 50. El píxel inicial asumió el valor 50. Previamente, cuando el píxel inicial tenía el valor=50, tomó el valor 100 del píxel de arriba (el filtro no trabaja sobre la imagen sino sobre una copia) y de esta manera desapareció en el fondo de píxeles a "100". Como resultado gráfico, el píxel inicial se movió un píxel hacia abajo.

Matriz kernel de 3x3: el valor se introduce directamente en las cajas.Divisor: El resultado del cálculo previo será dividido por este divisor. Difícilmente usará 1, que no variará el resultado, y 9 o 25 según el tamaño de la matriz, que da la media del valor del píxel.

Desplazamiento: este valor se suma al resultado de la división. Es útil si el resultado fuese negativo. Este desplazamiento puede ser negativo.

FIGURA 13

Ejemplo de aplicación del filtro de Sobel[3] consta de dos iteraciones de convolución horizontal y vertical para la detección efectiva de gradientes en imágenes no uniformes.

$$\mathbf{G_x} = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G_y} = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

[3] Filtro de Sobel: ejemplo e imágenes extraídas de: http://docs.gimp.org/es/plug-in-sobel.html

CAPÍTULO III

METODOLOGÍA

3.1 METODOLOGIA DE PROGRAMACION EXTREMA (XP)

3.1.1 ANALISIS

La metodología XP plantea en análisis como un permanente diálogo entre la parte empresarial y técnica del proyecto, en la que los primeros decidirán el alcance ¿Qué es lo realmente necesario del proyecto?, la prioridad qué debe ser hecho en primer lugar, la composición de las versiones que debería incluir.

3.1.2 DISEÑO

El Propósito del diseño es de crear una arquitectura para la naciente implementación, el diseño arquitectural sólo puede comenzar una vez que el equipo tenga un entendimiento razonable de los requerimientos del sistema. El diseño, como el análisis, nunca termina realmente hasta que el sistema final es entregado.

3.1.3 DESARROLLO

Esta etapa debe reunir las siguientes características o cualidades.

- El cliente está siempre disponible
- Se debe escribir código de acuerdo a los estándares
- Desarrollar la unidad de pruebas primero
- Todo el código debe programarse por parejas
- Integrar frecuentemente
- Todo el código es común a todos

3.1.4 PRUEBA

Todo el código debe ir acompañando, Los casos de prueba se escriben antes que el código. Los desarrolladores escriben pruebas unitarias y los clientes especifican pruebas funcionales.

3.2 MODELAMIENTO UML

3.2.1 LENGUAJE DE MODELAMIENTO UNIFICADO

UML (Unifed Modeling Language) es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos. Se ha convertido en el estándar de facto de la industria, debido a que ha sido impulsado por los autores de los tres métodos más usados de orientación a objetos: Grady Booch, Ivar Jacobson y Jim Rumbaugh. Estos autores fueron contratados por la empresa Rational Software Co. para crear una notación unificada en la que basar la construcción

de sus herramientas CASE[4]. En el proceso de creación de UML han participado, no obstante, otras empresas de gran peso en la industria como Microsoft, Hewlett-Packard, Oracle o IBM, así como grupos de analistas y desarrolladores, en un trabajo conjunto por la estandarización de una metodología y simbología de modelamiento.

Esta notación ha sido ampliamente aceptada debido al prestigio de sus creadores y debido a que incorpora las principales ventajas de cada uno de los métodos particulares en los que se basa (principalmente Booch, OMT y OOSE). UML ha puesto fin a las llamadas "guerras de métodos" que se han mantenido a lo largo de los 90, en las que los principales métodos sacaban nuevas versiones que incorporaban las técnicas de los demás. Con UML se fusiona la notación de estas técnicas para formar una herramienta compartida entre todos los ingenieros software que trabajan en el desarrollo orientado a objetos.

Uno de los objetivos principales de la creación de UML era posibilitar el intercambio de modelos entre las distintas herramientas **CASE** orientadas a objetos del mercado. Para ello era necesario definir una notación y semántica común. En la figura superior se puede ver cuál ha sido la evolución de UML hasta la creación de UML 1.3, en el que se basa este documento. Hay que tener en cuenta que el estándar UML no define un

[4]CASE : Computer Aided Software Engineering (Ingenieria de Software Asistido por Computadora)

proceso de desarrollo específico, tan solo se trata de una notación. En estos conceptos se sigue el método propuesto por **Craig Larman**.

3.2.2 DIAGRAMAS DE CASOS DE USO

El caso de uso es una excelente herramienta para estimular a que los usuarios hablen, de un sistema, desde sus propios puntos de vista. No siempre es fácil para los usuarios explicar cómo pretenden utilizar un sistema.

La idea es involucrar a los usuarios en las etapas iníciales del análisis y diseño del sistema. Esto aumenta la probabilidad de que el sistema sea de mayor provecho para la gente a la que ayudara, en lugar de ser un manojo de expresiones de computación incomprensibles e inmanejables por los usuarios finales.

3.2.3 DIAGRAMAS DE SECUENCIA

Permite plasmar es una secuencia grafica los diferentes pasos a realizar para la obtención de un proyecto, no tendremos mayores problemas con el nuestro puesto que la mayoría de estos casos tiene que ser documentada.

3.2.4 DIAGRAMAS DE FLUJO

Aquí podremos graficar la secuencia de acciones y sub-acciones requeridas bajo condiciones o bucles de iteración repetitiva, en fin una secuencia grafica de ejecución, para nuestro caso particular

de la automatización de la remasterización de la Imagen ISO original hasta la instalación de nuevos paquetes, configuraciones varias y finalmente la generación de la Imagen ISO final y la puesta en prueba de esta.

3.3 METRICA DE VALIDACION ISO 9126

ISO 9126 es un estándar internacional para la evaluación de la calidad del software. Está reemplazado por el proyecto SQuaRE, ISO 25000:2005, el cual sigue los mismos conceptos. Este estándar es el más usado... El estándar está dividido en cuatro partes las cuales dirigen, realidad, métricas externas, métricas internas y calidad en las métricas de uso y expendido. El modelo de calidad establecido en la primera parte del estándar, ISO 9126-1, clasifica la calidad del software en un conjunto estructurado de características y sub-características.

3.3.1 FUNCIONALIDAD

Un conjunto de atributos que se relacionan con la existencia de un conjunto de funciones y sus propiedades específicas. Las funciones son aquellas que satisfacen las necesidades implícitas o explícitas.

- **Idoneidad** - Atributos del software relacionados con la presencia y aptitud de un conjunto de funciones para tareas especificadas.
- **Exactitud** - Atributos del software relacionados con la disposición de resultados o efectos correctos o acordados.

- **Interoperabilidad** - Atributos del software que se relacionan con su habilidad para la interacción con sistemas especificados.
- **Seguridad** - Atributos del software relacionados con su habilidad para prevenir acceso no autorizado ya sea accidental o deliberado, a programas y datos.
- Cumplimiento de normas.

3.3.2 FIABILIDAD

Un conjunto de atributos relacionados con la capacidad del software de mantener su nivel de prestación bajo condiciones establecidas durante un período establecido.

- **Madurez** - Atributos del software que se relacionan con la frecuencia de falla por fallas en el software.
- **Recuperabilidad** - Atributos del software que se relacionan con la capacidad para restablecer su nivel de desempeño y recuperar los datos directamente afectos en caso de falla y en el tiempo y esfuerzo relacionado para ello.
- **Tolerancia a fallos** - Atributos del software que se relacionan con su habilidad para mantener un nivel especificado de desempeño en casos de fallas de software o de una infracción a su interfaz especificada.
- **Usabilidad** - Un conjunto de atributos relacionados con el esfuerzo necesario para su uso, y en la valoración individual de tal uso, por un establecido o implicado conjuntos.

- **Aprendizaje** - Atributos del software que se relacionan al esfuerzo de los usuarios para reconocer el concepto lógico y sus aplicaciones.
- **Comprensión** - Atributos del software que se relacionan al esfuerzo de los usuarios para reconocer el concepto lógico y sus aplicaciones.
- **Operatividad** - Atributos del software que se relacionan con el esfuerzo de los usuarios para la operación y control del software

3.3.3 EFICIENCIA

Conjunto de atributos relacionados con la relación entre el nivel de desempeño del software y la cantidad de recursos necesitados bajo condiciones establecidas.

- Comportamiento en el tiempo - Atributos del software que se relacionan con los tiempos de respuesta y procesamiento y en las tasas de rendimientos en desempeñar su función.
- Comportamiento de recursos.

3.3.4 MANTENIBILIDAD

Conjunto de atributos relacionados con la facilidad de extender, modificar o corregir errores en un sistema software.

- Estabilidad - Atributos del software relacionados con el riesgo de efectos inesperados por modificaciones.

- Facilidad de análisis - Atributos del software relacionados con el esfuerzo necesario para el diagnostico de deficiencias o causas de fallos, o identificaciones de partes a modificar.

- Facilidad de cambio - Atributos del software relacionados con el esfuerzo necesario para la modificación, corrección de falla, o cambio de ambiente.

- Facilidad de pruebas - Atributos del software relacionados con el esfuerzo necesario para validar el software modificado.

3.3.5 PORTABILIDAD

Conjunto de atributos relacionados con la capacidad de un sistema software para ser transferido desde una plataforma a otra.

- Capacidad de instalación - Atributos del software relacionados con el esfuerzo necesario para instalar el software en un ambiente especificado.

- Capacidad de reemplazamiento - Atributos del software relacionados con la oportunidad y esfuerzo de usar el software en lugar de otro software especificado en el ambiente de dicho software especificado.

- Adaptabilidad - Atributos del software relacionados con la oportunidad para su adaptación a diferentes ambientes

especificados sin aplicar otras acciones o medios que los proporcionados para este propósito por el software considerado.

- Co-Existencia

- Factores (especificar): Describen la visión externa del software, como es visto por los usuarios.

- Criterios (construir): Describen la visión interna del software, como es visto por el desarrollador.

- Métricas (controlar): Se definen y se usan para proveer una escala y método para la medida.

ISO 9126 distingue entre fallo y no conformidad. Un fallo es el incumplimiento de los requisitos previos, mientras que la no conformidad es el incumplimiento de los requisitos especificados. Una distinción similar es la que se establece entre validación y verificación.

CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

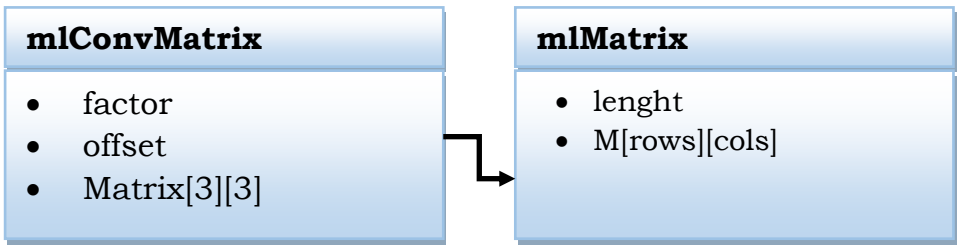
4.1 MODELAMIENTO

4.1.1 DIAGRAMA DE CLASES

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, orientados a objetos.el diagrama de clases puede tener como ejemplo: una clase que sería un objeto o persona misma en la cual se especifica cada acción y especificación.

Describimos el contenido y las especificaciones de las clases implementadas para el Framework.

CUADRO 1
Descripción y detalle del diagrama de clases mlConvMatrix y mlMatrix



Clase: mlMatrix

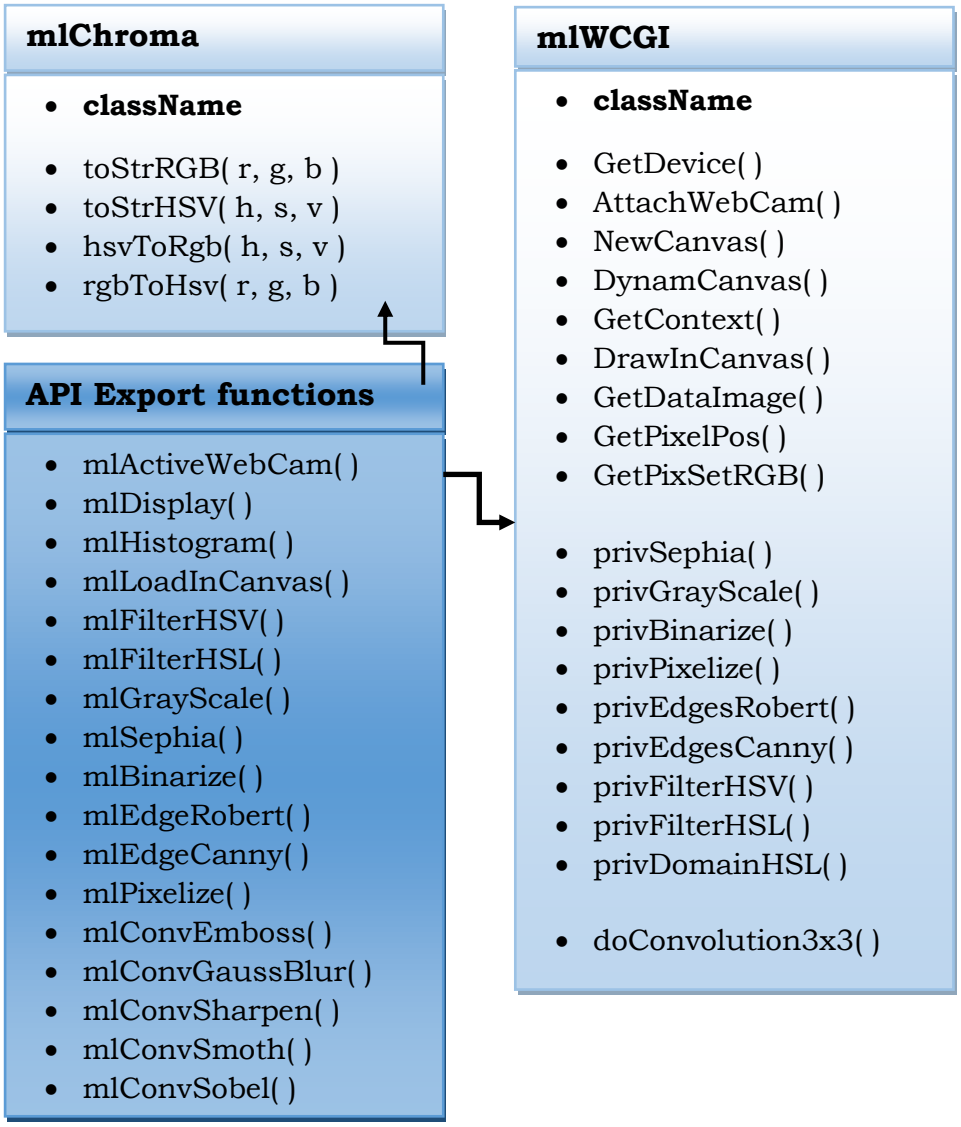
Esta clase se ha definido para la creación dinámica de matrices de dimensión también dinámica, para las operaciones de Convolución, que se usaran en las clases posteriores.

Clase: mlConvMatrix

Esta clase se ha definido para soportar la matriz Kernel de 3x3, se requiere variables como **offset** y **factor** para realizar la operación y se asignan a cada Kernel por ello la necesidad de declararse en una clase que integre los elementos mencionados.

CUADRO 2

Descripción y detalle del diagrama de las clases mlChroma y mlWCGI.



Clase: mlWCGI

En esta clase se definen las funciones de conversión de los componentes de ingreso de datos, manipulación de <IMG>, <Canvas>, <Video> y <UserMedia> para la WebCam, los grupos de funciones:

- Funciones de activación de Dispositivos, conversión y uniformización de entradas para procesamiento y recuperación de mapa de bits y localización de coordenadas de pixeles.
- Rutinas privadas para la aplicación de filtros y efectos fotográficos.
- Rutinas de operación de convolución de matrices. (nótese la simplificación, solo requiere dos argumentos el mapa de bits uniformizado y la Matriz Kernel para realizar la convolución)

El Framework basa su colección de funciones API descrito en el bloque (API Export Functions) que simplifica el acceso a métodos de procesamiento de imágenes, video o entrada de WebCam adquiridos desde cualquier dispositivo de entrada HTML5, como <IMG>, <CONVAS> o <VIDEO> para evitar la tediosa verificación manual del dispositivo de entrada las APIs encapsulan el proceso de verificación y adquisición obteniendo una manipulador genérico para ser re-eviados por las funciones de exportación hacia la clase que hace el procesamiento de la entrada, es decir la clase ***mlWCGI.***

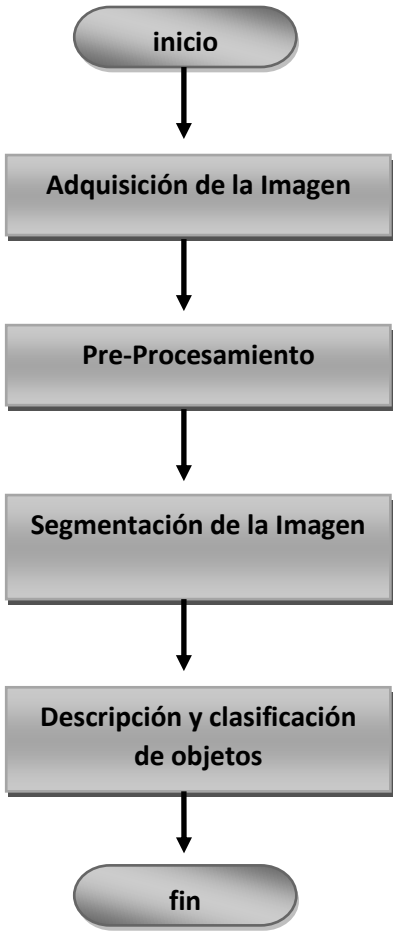
La clase ***mlWCGI***, esta implementada pensado en una entrada genérica de mapa de bits, no importando el formato de entrada pues ese proceso lo realizan las funciones API, esta clases realiza los cálculos mediante los algoritmos adaptados para el procesamiento de imágenes y obtención de características, una vez concluidos estos son re-enviados al usuario.

4.1.2 DIAGRAMAS DE FLUJO

- Adquisición de la imagen.
- Pre-procesamiento.
- Segmentación de la imagen.
- Descripción y clasificación de objetos.

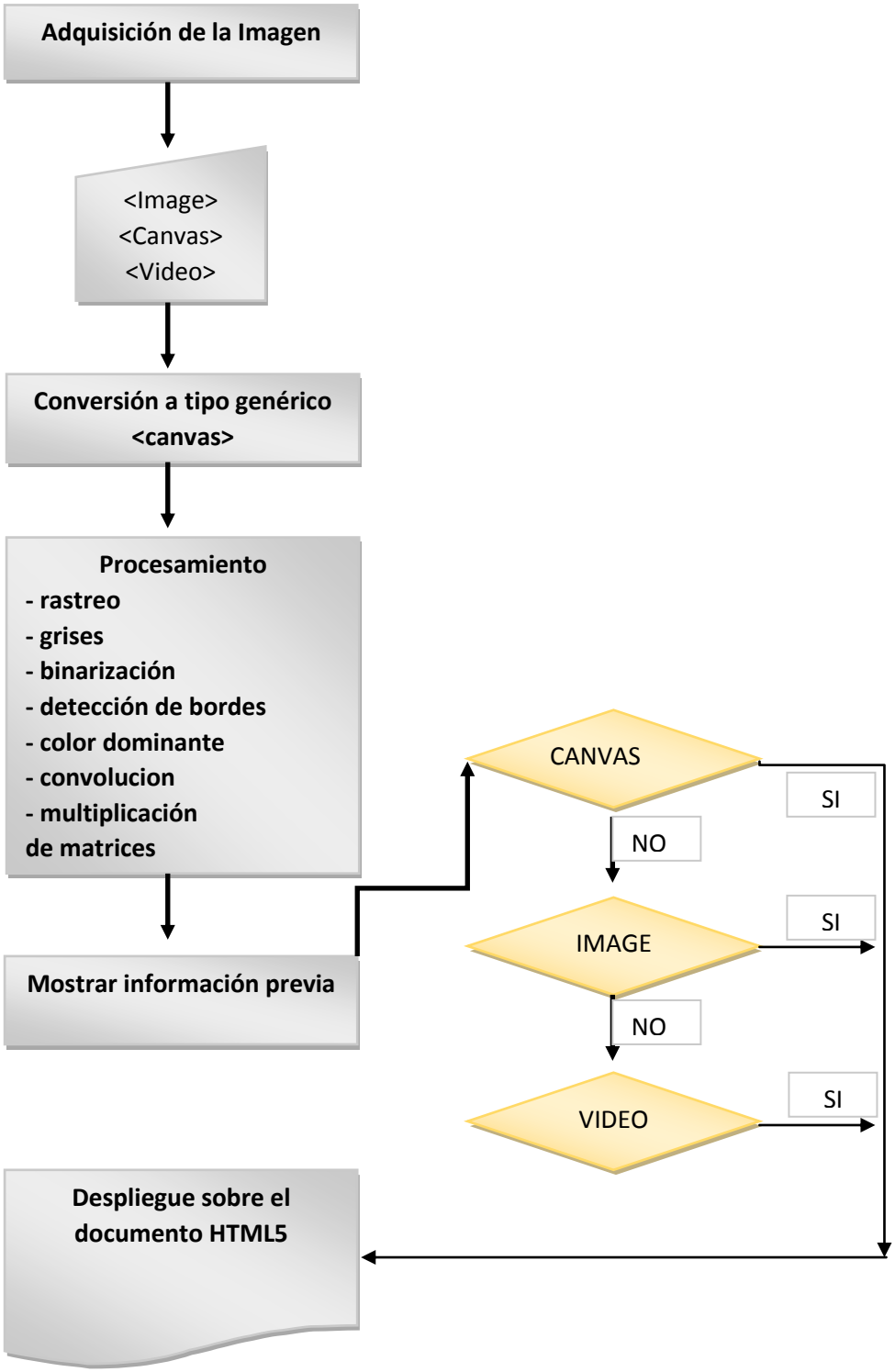
CUADRO 3

Representación de la secuencia del procesamiento digital de imágenes, el cual será adaptado para el Framework Web.



CUADRO 4

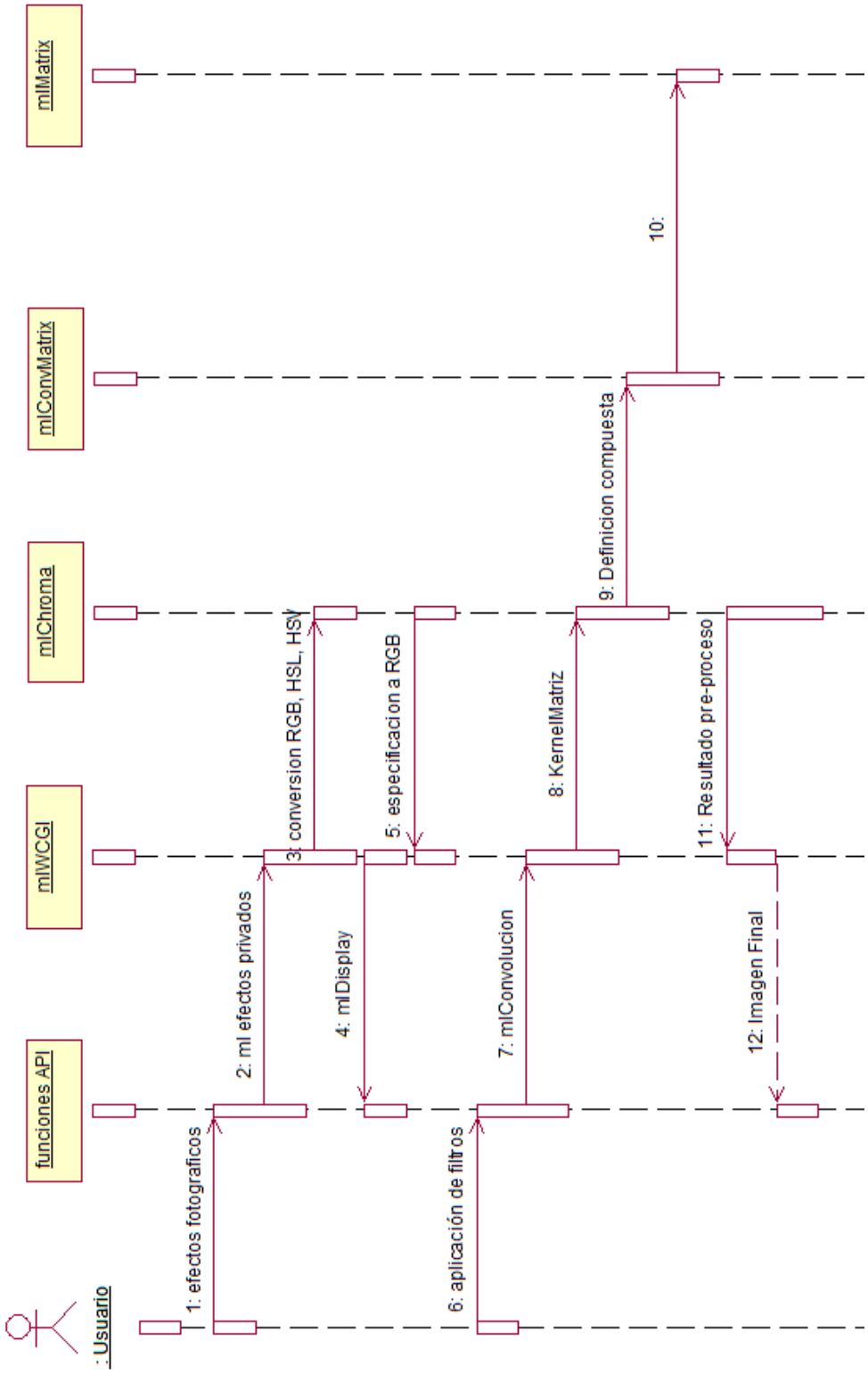
Representación del flujo de adquisición y devolución de imágenes sobre los controles para HTML5.



4.1.3 DIAGRAMA DE SECUENCIA

CUADRO 5

Diagrama de Secuencia de procesos de aplicación de efectos y filtros



4.2 ANÁLISIS DE REQUERIMIENTOS

4.2.1 SOFTWARE A UTILIZAR

Para no tener problemas con las licencias y derechos sobre el desarrollo de los módulos y las pruebas del mismo, durante el desarrollo, implementación y depuración.Se ha usado Software Libre basado en la licencia GPL.

- SublimeText[5] 3.0 - Editor de código HTML, CSS3, Javascript.
- GIMP – Software de manipulación de imágenes.
- SciLab 5.5.1 – Paquete extra SIVP - Scilab Image and Video Processing Toolbox.
- Google Chrome y Mozilla Firefox – Navegadores Web con soporte HTML5.
- MonoDeveloper C# - Ejecutar ejemplos antes de convertirlos a lenguaje Javascript.
- Editor TextMaker y OpenOffice Write – Procesador de textos y editor LaTeX para las diapositivas Beamer[6] LaTeX.
- Servidor LAMP[7] – Apache.
- Sistema Operativo GNU/Linux Kubuntu[8] 14.04

[5]SublimeText – Este software no es del todo gratuito pues viene en un periodo de prueba, pero se permite su uso sin registrar formalmente, no hay penalidad.

[6]Beamer – Es un paquete extra para LaTeX para editar presentaciones profesionales el nombre proviene de la lengua germana Beamer (Transparencias)

[7] LAMP – Colección de Linux – Apache – MySQL – PHP.

[8]Kubuntu – Esta basado en Linux Ubuntu con escritorio KDE permitido sin pago de licencia.

4.3 IMPLEMENTACIÓN DE MÓDULOS

4.3.1 ESPECIFICACIÓN DE CÓDIGO BASE HTML5

Código fuente HTML básico para permitir el uso del Framework en modo implementación y/o modo de modificación.

```
1  <!DOCTYPE html>
2  <HTML lang="es">
3  <head>
4  <meta charset="utf-8">
5  <title> WebCGI Examples </title>
6  <script src="./JS/mlCGI.e.js"></script>
7  <script src="./JS/mlDialogs.js"></script>
8  <link href="./CSS/mlstyle.css" rel="stylesheet">
9  </head>
10
11 <BODY>
12 <div> Hi! </div>
13 </BODY>
14 </HTML>
```

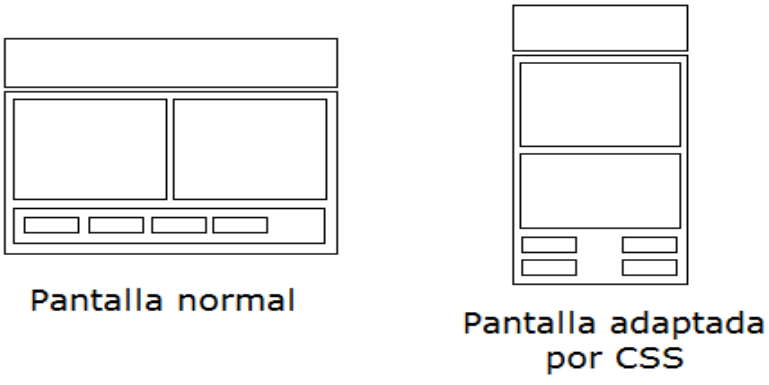
4.3.2 AGREGANDO EL DISEÑO RESPONSIVO

El diseño web adaptable (en inglés, Responsive Web Design) es una filosofía de diseño y desarrollo web que mediante el uso de estructuras e imágenes fluidas, así como de “media-queries” en la hoja de estilo CSS, consigue adaptar el sitio web al entorno del usuario. El diseñador y autor norteamericano Ethan Marcotte. ***(Marcotte, 2012)*** creó y difundió esta técnica a partir de una serie de artículos en A List Apart, una publicación en línea especializada en diseño y desarrollo web, idea que luego extendería en su libro Responsive Web Design.

Código fuente CSS usado para modificar el logo del Framework una vez que cambie la resolución del browser (El contenido no depende de la resolución en pantalla del dispositivo activo sino de las dimensiones del Navegador)

```
1
2  /* viewWidt < 320  : Smartphone */
3  @media only screen and (max-width:320px){
4        body{
5              background-color: red;
6        }
7
8        #foto{
9              height: 115px;
10             border: 1px solid black;
11             background-image: url("graficoS.jpg");
12       }
13 }
14
15 /* viewWidt > 320 && viewWidth <480 : Smartphone
16 Landscape */
17 @media only screen and (min-width: 320px)
18                     and (max-width: 480px) {
19       body{
20             background-color: green;
21       }
22       #foto{
23             height: 115px;
24             float: left;
25             border: 1px solid black;
26             background-image: url("graficoL.jpg");
27       }
28 }
```

FIGURA 14
El código fuente escrito en el archivo *mlstyle.css* adaptará los controles en la forma detallada en el gráfico.



4.3.3 SOPORTE DE ENTRADA Y SALIDA MULTIMEDIOS

- **<IMG>**Esta etiqueta permite mostrar una imagen en cualquier formato, el Navegador ya integra el soporte para los tipos de imagen BMP, JPG, PNG y GIF, este control permite mostrar la imagen en modo ajustado (stretch) para cuando el tamaño real de la imagen no pueda mostrarse sobre el Navegador, se le asigna una dimensión más pequeña. La forma de declarar un identificador a esta etiqueta es:

```
HTML
<img id=”fotos” src=”” width=300 height=250>

Javascript
imFoto = document.getElementById(“fotos”);
```

- **<CANVAS>**Esta nueva etiqueta de HTML5, supera a la anterior etiqueta al incluir soporte para dibujar y desplegar gráficos en 2D y 3D soportando WebGL (OpenGL para entornos Web), pero el punto más importante es su API de desarrollo que permite escribir cualquier primitiva geométrica, desplegar imágenes y acceder al mapa de bits en memoria. La forma de declarar un identificador a esta etiqueta es:

```
HTML
<canvas id=”lona” width=350 height=250></canvas>

Javascript
cnvs = document.getElementById(“lona”);
cntx = cnvs.getContext(“2D”);

   cntx.fillStyle = ”RGB(0,255,0)”;
cntx.fillRect(0,0,320,240);
```

- **<VIDEO>**Esta nueva etiqueta de HTML5, permite reproducir un archivo en formato de video, los tipos soportados son ***mp4, ogg y webm***, esta etiqueta actúa como un canal de streaming de video, por lo que soporta la reproducción de una WebCam, incrementando su funcionalidad. La forma de declarar un identificador para esta etiqueta es:

```
HTML
<video id="video" width=350 height=250>
<source src="demo.mp4" type="video/mp4">
</video>

Javascript
vid = document.getElementById("video");

cam = navigator.getUserMedia( {video:true},
function ( stream ){
vid.src = URL.createObjectURL(stream);
vid.play();
          },
function ( err ){
console.log("error en webCam");
          }
        );
```

Uniformizar los distintos tipos de entrada para que reciban cualquier efecto o filtro de la clase mlWCGI, se emplea un Canvas en memoria y creando una copia para enviar al pre-procesador y devolver un mapa de bits.

```
Javascript:
function mlFilterX( idSource ){
     // crea una copia en memoria volcando contenido
canv = wcgi.DynamCanvas( idSource );

wcgi.privFilterX( canv ); // procesar la copia

     // finalmente sobre-escribir la copia procesada
mlDisplay( canv, idSource );
}
```

4.3.4 IMPLEMENTACIÓN DE CLASES EN JAVASCRIPT

Javascript tiene implementada la asignación dinámica de objetos singulares, aún no soporta la herencia o polimorfismo (como es el caso de Java ó C#) pero es posible definir variables y funciones miembro los que etiquetados como un objeto, estos son instanciables una o varias veces.Definición de las clases implementadas para el Framework, en las líneas finales la instanciación de objetos a usar por las API de exportación.

```
1
2  /* clase: matriz de convolucion */
3  function mlConvMatrix( ) {
4        this.factor = 1;
5        this.offset = 3;
6        this.M = [
7              [1, 1, 1],
8              [1, 1, 1],
9              [1, 1, 1],
10       ];
11 }
12
13 /* clase: conversión de espacios de color */
14 function mlChroma( className ) {
15
16       this.mClassName = className;
17       this.toStrRGB = function ( r, g, b ){
18             return "RGB(r,g,b)";
19       }
20 };
21
22 /* clase: manipulación de mapa de bits */
23 function mlWCGI( className ) {
24
25       this.mClassName = className;
26
27       this.GetDevice = function ( idDev ){
28       returnnavigator.getUserMedia;
29       }
30 );
31
32 /* instanciación de objetos */
33 var wcgi = new mlWCGI("t1");
34 var wrgb = new mlChroma("t1");
35
```

4.3.5 DETECCIÓN DE DISPOSITIVO MULTIMEDIA

Función miembro que permite detectar el tipo de dispositivo y determinar el destino.

```
this.GetDevice = function ( idDev )
{
      if( idDev == "[object HTMLVideoElement]" )
            return ML_DEV_VIDEO;
      else if( idDev == "[object HTMLImageElement]" )
            return ML_DEV_IMAGE;
      else if( idDev == "[object HTMLCanvasElement]" )
            return ML_DEV_CANVAS;
      else{
            alert( "ClassError: Object Undefined" );
            return ML_DEV_NULL;
      }
}
```

Función API que desplegará la información pre-procesada de la forma adecuada una vez determinado el dispositivo multimedia.

```
function mlDisplay( idSource, idDestin ){

// all media draws it over canvas
if( wcgi.GetDevice(idDestin) == ML_DEV_CANVAS ){
wcgi.DrawInCanvas( idDestin, idSource );
}
// conversion media over img
else if( wcgi.GetDevice(idDestin) == ML_DEV_IMAGE ){

if( wcgi.GetDevice(idSource) == ML_DEV_IMAGE ){
idDestin.src = idSource.src;
}
if( wcgi.GetDevice(idSource) == ML_DEV_VIDEO ){
var canv = wcgi.DynamCanvas(idSource);
idDestin.src = canv.toDataURL();
}
if( wcgi.GetDevice(idSource) == ML_DEV_CANVAS ){
idDestin.src = idSource.toDataURL();
}
}
else{
alert("Can't write Output");
}
}

```

4.3.6 CORRECCIÓN DE TAMAÑO REAL Y VISUAL

Debemos tener en cuenta que la imagen sea la dimensión que tengo, no podrá ser mostrada en su real dimensión por cuestiones estéticas, esto no debe afectar el procesamiento y obtención de resultados, para mostrar el problema de dimensiones se tiene este ejemplo para mostrar el problema de las dimensiones de imagen control Canvas y CSS.

CUADRO 6

Dimensiones obtenidas de la imagen, control Canvas y CSS

	Ancho	Alto
Imagen JPG, PNG	**800**	**520**
Canvas	**420**	**270**
CSS Canvas	**300**	**150**
Vista optima	**Canvas**	**Canvas**

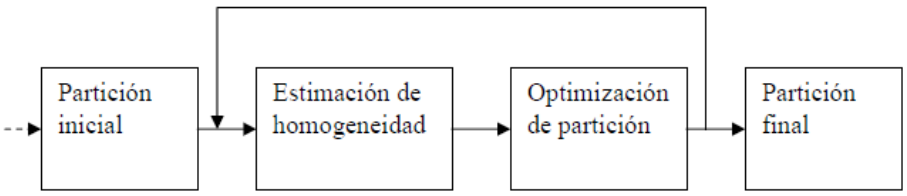
FIGURA 15

Para la corrección de las dimensiones puede observarse en la tabla previa que debe asignarse el CSS con las mismas dimensiones del control Canvas y para la imagen se tendrá una copia en memoria, pero al desplegar se realizara un escalado o stretch.

4.3.7 DISCRETIZACION DEL COLOR

La aplicación de filtros, efectos y seleccionadores será representado por el diagrama de la figura 14, mostrando las etapas subjetivas hasta la finalización y obtención del resultado esperado.

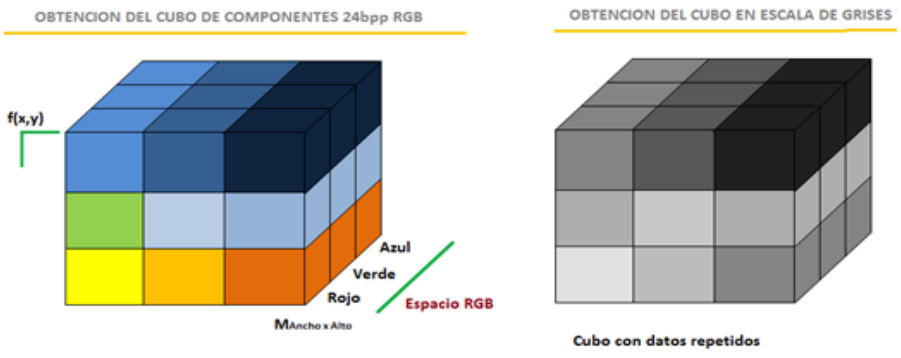
FIGURA 16
Esquema general del algoritmo de discretización.



La discretización y/o cuantificación del contenido de las imágenes de entrada se almacenaran sobre matrices tridimensionales de valores que contendrán tonos de color RGB, incluso si esta en tonos de grises o binarizado.

```
Javascript
var pixmap = dataIm.data;
for( var i=0; i<pixmap.length; i+=4 ) // RGBA
{
pixmap[i+0] = 0;
}
```

FIGURA 17
Distribución RGB de matrices tridimensionales sobre imágenes en True Color 24bits, escala de grises de 8bits incluso si se presenta como monocromática



4.3.8 CONVERTIR A ESCALA DE GRISES

Una escala de grises es una escala en la que el valor de cada píxel posee un valor, una graduación de gris de 0 - 255. Lasimagen ingresa con tres tonos de color sacando una media aritmética, lograremos el tono de gris deseado. Procedemos con el código:

```
this.privGrayScale = function( idCanvas ) {
      // obtención del contexto
      var pixmap = dataIm.data;

      for( var i=0; i<pixmap.length; i+=4 ) // RGBA
      {
            varmean = (red + green + blue) / 3;
            pixmap[i+0] =
            pixmap[i+1] =
            pixmap[i+2] = mean;
      }
}
```

FIGURA 18
Imagen obtenida en grises, se muestra el histograma RGB de entrada y una vez procesado se muestra el nuevo histograma de la imagen.

4.3.9 BINARIZACIÓN POR UMBRAL (THRESHOLD)

Previamente se habrá convertido a escala de grises y con 256 tonos de grises y bajo un criterio de discriminación conocido como umbral (threshold) (0< x < 255)estableceremos entre blanco los valores menores y negro los valores mayores o iguales. Procedemos con el código:

```
this.privBinarize = function( idCanvas,Threshold){
      // obtención del contexto
      var pixmap = dataIm.data;

      for( var i=0; i<pixmap.length; i+=4 ) // RGBA
      {
            varmean = (red + green + blue) / 3;
            varcolor = (mean > Threshold)? 255 : 0;
            pixmap[i+0] =
            pixmap[i+1] =
            pixmap[i+2] = color;
      }
`
```

FIGURA 19

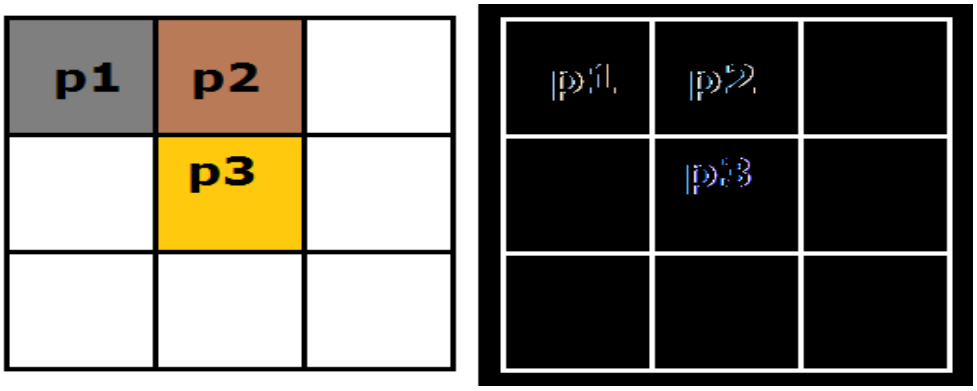
(arriba) imagen obtenida al binarizar con un umbral de 127. (abajo) imagen obtenida con un umbral de 200. Modificando el umbral se obtiene una mejor tonalidad para lograr fronteras o bordes.

4.3.10 ADAPTACIÓN PARA DETECCION DE BORDES

Se ha optado por la técnica de detección de bordes de Robert[9], detección por gradiente al tener que procesar imágenes con superficies no uniformes. Comparado con la detección por Sobel o Canny, que requieren dos iteraciones horizontal y vertical, esta técnica solo requiere una pasada al optimizar un doble barrido en la misma iteración. El código adaptado tendrá la forma:

```
this.privEdgeRober = function( idCanvas ) {
      // obtención del contexto
      var pixmap = dataIm.data;

      for (y = 0; y < idCanvas.height; y++)
      for (x = 0; x < idCanvas.width; x++) {
            p1 = this.getPixelPos(x+0, y+0 );
            p2 = this.getPixelPos( x+1, y+0 );
            p3 = this.getPixelPos( x+1, y+1 );

            cR = Math.pow( p1-p2, 2 );
            nPix1  = Math.sqrt(cR + cG + cB);

            cR = Math.pow( p1 - p3, 2 );
            nPix2  = Math.sqrt(cR + cG + cB);

            nPixel = ( nPix1>=Threshold ||
                       nPix2>=Threshold )? 255:0;
            pixmap[ p1+ RGB ] = nPixel;
      }
}
```

FIGURA 20
Esquema de funcionamiento de la detección de Robert y ejemplo B/N

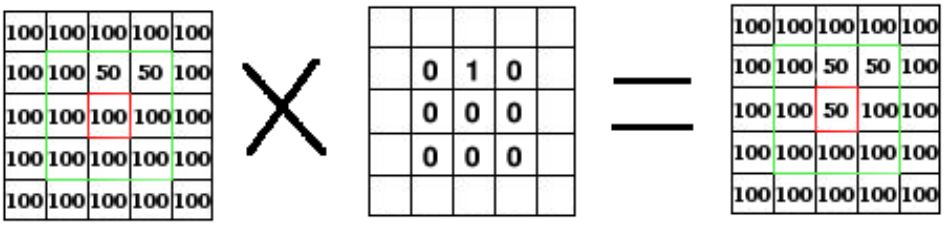


[9] Robert Lawrance : http://en.wikipedia.org/wiki/Roberts_cross

4.3.11 ADAPTACIÓN PARA LA CONVOLUCIÓN

Convolución es un operador matemático que transforma dos funciones **f** y **g** en una tercera función. Representa la magnitud en la que se superponen **f** y una versión trasladada e invertida de **g**. Una convolución es un tipo muy general de media móvil.

$$F(x,y) = f(x,y) * g(x,y) = \sum_i \sum_j f(x+i, y+j) h(i,j)$$



$$r = (100*0 + 50*1 + 50*0) + (100*0 + 100*0 + 100*0) + 100*0 + 100*0 + 100*0$$
$$r = 50 + 0 + 0$$
$$r = 50$$

Donde r será asignado a la posición M[2][2], para este resultado se han tomado lo elementos de los vecinos con la siguiente distribución de matrices.

```
r = M[1][1]*K[1][1]+M[1][2]*K[1][2]+M[1][3]*K[1][3] +
    M[2][1]*K[2][1]+M[2][2]*K[2][2]+M[1][2]*K[1][2] +
M[3][1]*K[3][1]+M[3][2]*K[3][2]+M[3][3]*K[3][3] ;

r = x
R[2][2] = x
```

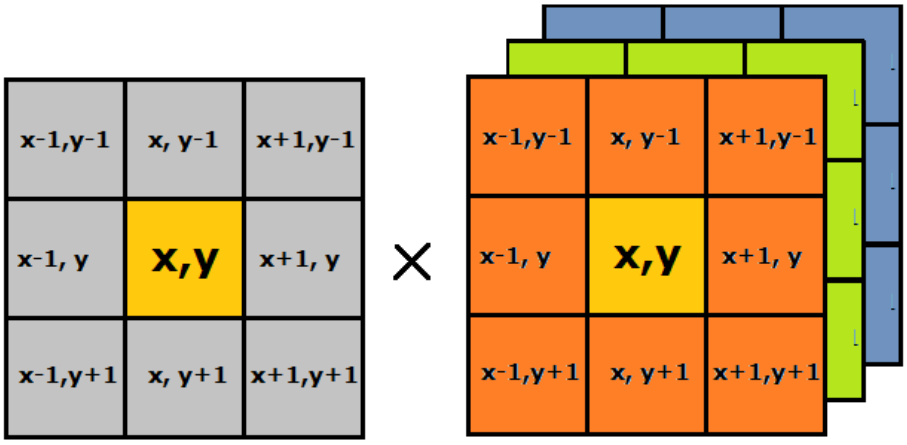
Pero debemos tener en cuenta que las imágenes están compuestas de 3 juegos de matrices, tono rojo, verde y azul.

Un detalle a tener en cuenta para la implementación es que las imágenes tienes 3 matrices (RGB), por lo que el proceso tomara 3 iteraciones para obtener el color transformado final. Se procede con la declaración de la clase ***Matriz de Convolución***.

```
function mlConvMatrix( ) {
      this.factor = 1;
      this.offset = 3;
      this.M = [
            [1, 1, 1],
            [1, 1, 1],
            [1, 1, 1],
      ];
}

var kerConv = new mlConvMatrix();
var canvas = wcgi.DynamCanvas(idSource);

kerConv.offset = 0;
kerConv.factor = 1;
kerConv.M = [ [ 1, 0, -1 ],
            [ 2, 0, -2 ],
            [ 1, 0, -1 ], ];

wcgi.doConvolution3x3( canvas, kerConv );
```

FIGURA 21

Representación de la convolución por matrices matriz Kernel por cada uno de las matrices R,G, B las que deberán ser recompuestas para la imagen final reemplazando el contenido total.



```
// matrix sweep, edges except
for (y = 1; y < idCanvas.height-1; y++) {
      for (x = 1; x < idCanvas.width-1; x++) {

      pixRGBA = [
            [ (x-1,y-1), (x,y-1), (x+1,y-1) ],
            [ (x-1,  y), ( x, y), (x+1, y ) ],
            [ (x-1,y+1), (x,y+1), (x+1,y+1) ],
      ];

      for( i=ML_RED; i<=ML_BLUE; i++ ){

      pixRes =
            pixRGBA[0][0][i] * Ker.M[0][0] +
            pixRGBA[0][1][i] * Ker.M[0][1] +
            pixRGBA[0][2][i] * Ker.M[0][2] +
            pixRGBA[1][0][i] * Ker.M[1][0] +
            pixRGBA[1][1][i] * Ker.M[1][1] +
            pixRGBA[1][2][i] * Ker.M[1][2] +
            pixRGBA[2][0][i] * Ker.M[2][0] +
            pixRGBA[2][1][i] * Ker.M[2][1] +
            pixRGBA[2][2][i] * Ker.M[2][2] ;

            pixRes = (pixRes/Ker.factor)+Ker.offset;

            if (pixRes < 0)   pixRes = 0;
            if (pixRes > 255) pixRes = 255;

            pixmap[ offset+i ] = pixRes;
         }
      }
}
```

4.3.12 LISTA DISPONIBLE DE FILTROS POR CONVOLUCIÓN

Una vez implementado el modulo para generar la convolución, se optimiza la función para que sea capaz de operar con matrices personalizables y enumeramos los filtros más usados y generarlos como una función para no tener que implementar como usuario sino directamente usar la función API.

- Filtro para desenfoque (Blur)

$$matConv = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- Filtro para Desenfoque Gaussiano (Gaussian Blur)

$$matConv = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 8 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

- Filtro de suavizado (Smoth)

$$matConv = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 4 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- Filtro de realce (Emboss)

$$matConv = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

- Filtro de definición (Sharpness)

$$matConv = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 10 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

- Filtro de definición por media (Mean Removal)

$$matConv = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

- Filtro para detección de bordes

$$matConv = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

- Filtro de Sobel - Horizontal y vertical

$$matConv \;:\; H = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad V = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Una vez definidos los valores de las matrices kernel, se procede a escribir el código fuente que permita aplicar el filtro sobre el contenido de la imagen y para poder visualizarla tenemos:

```
function mlConvEdgeSobel( idSource, idDestin ) {

      var kerConv = new mlConvMatrix();
      var canv = wcgi.DynamCanvas(idSource);

      // primero uniformizar los tonos RGB
      wcgi.privGrayScale( canv );

      // convolución vertical
      kerConv.M  =  [ [ 1, 0, -1 ],
                      [ 2, 0, -2 ],
                      [ 1, 0, -1 ], ];
      wcgi.doConvolution3x3( canv, kerConv );

      // convolución horizontal
      kerConv.M =[ [  1,  2,  1 ],
                   [  0,  0,  0 ],
                   [ -1, -2, -1 ], ];
      wcgi.doConvolution3x3( canv, kerConv );
      mlDisplay( canv, (idDestin)?idDestin:idSource
);
```

4.4 ANÁLISIS Y PRUEBAS

4.4.1 INSTALACIÓN DEL SERVIDOR WEB PARA PRUEBAS

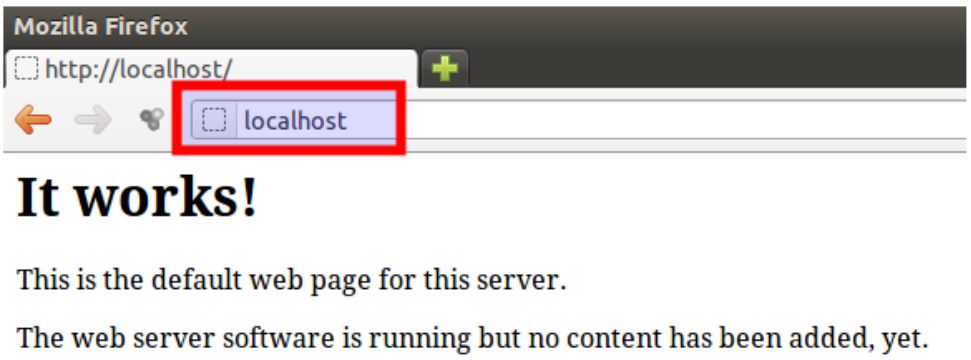
Se ha trabajado sobre el sistema operativo GNU/Linux Kubuntu 11.04, para la implementación de los módulos se ha instalado LAMP (Apache, MySQL, PHP) usando las sentencias:

```
$ sudo apt-get update
$ sudo apt-get install apache2 php5
```

Una vez instalado se procede a iniciar el servidor, para ver su funcionamiento abra una ventana en el navegador y escriba: http://localhost

```
$ chdir /var/www/html
$ sudo /etc/init.d/apache2 start
```

FIGURA 22
Navegador mostrando el funcionamiento del servidor Apache local, ya podemos implementar y probar los módulos del Framework.



4.4.2 INSTALACIÓN DEL EDITOR SUBLIME TEXT

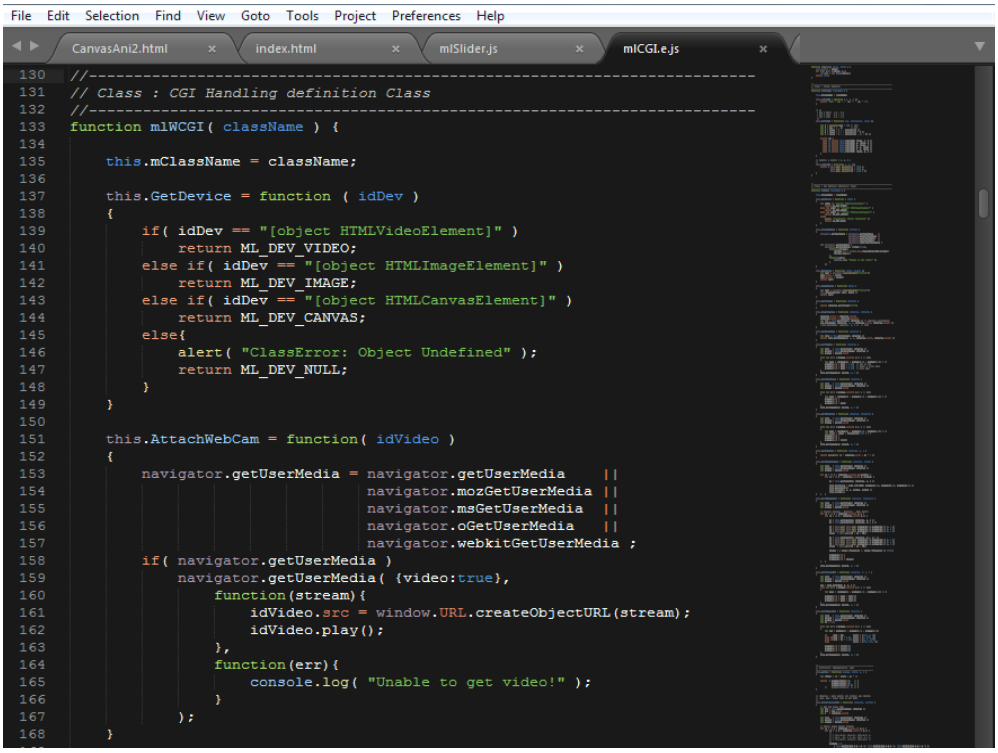
SublimeText es un editor rápido, sencillo e intuitivo de usar, la ayuda de sugerencia de código (code-suggest) acelera la escritura de módulos en CSS3 y Javascript.

Por defecto este editor no está incluido en los repositorios de Ubuntu, así que tendremos que agregarlo manualmente para proceder con su instalación por repositorio actualizado.

```
$ sudo add-apt-repository ppa:webupd8team/sublime-text-3

$ sudo apt-get update
$ sudo apt-get install sublime-text-installer
```

FIGURA 23

Editor Sublime con el código fuente de la clase mlWCGI, modulo base de nuestro Framework, detallando dos funciones miembro GetDevice y AttachWebCam, con ayuda de resaltado de código y code-suggest.

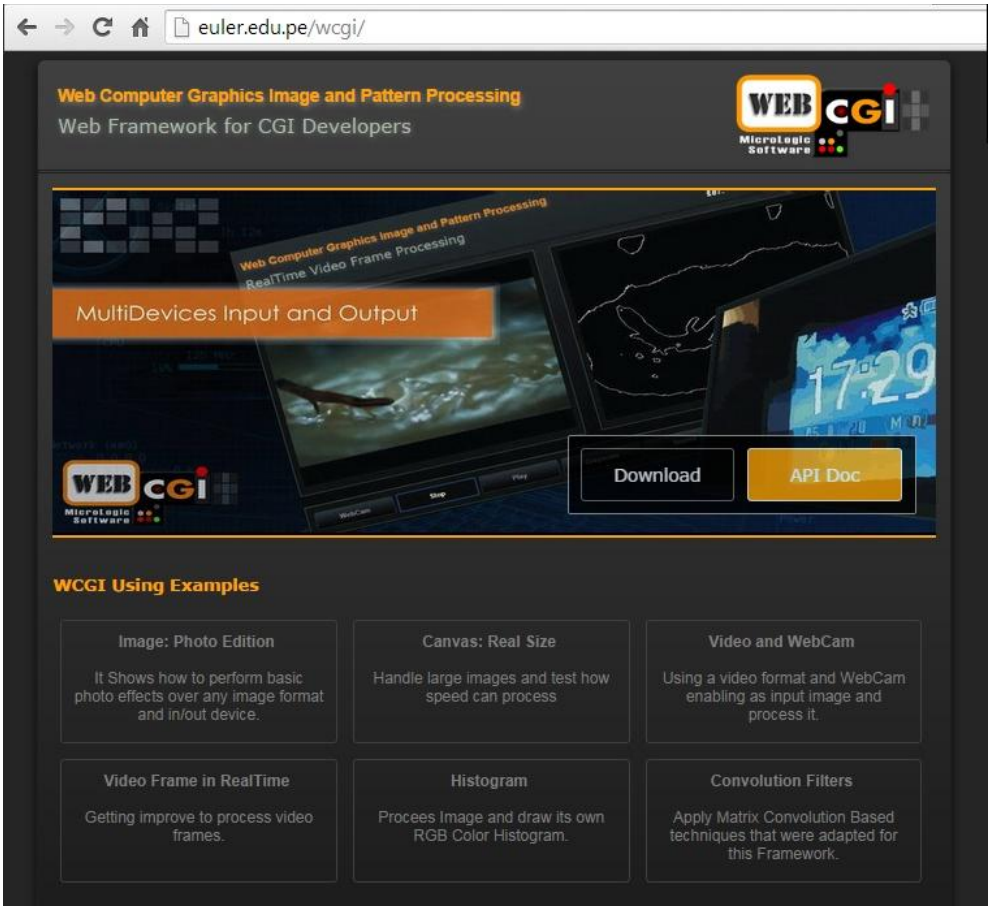


4.4.3 DOCUMENTACIÓN PÚBLICA ON-LINE

Una vez concluida la implementación de las rutinas del Framework, se procede con documentar el API de las funciones de exportación y como tal agregar ejemplos demostrativos para mostrar su funcionalidad y hacerlo accesible para nuevos desarrolladores, para los fines demostrativo se ha subido la información del Framework en el sitio web: http://euler.edu.pe/wcgi, al ingresar se tiene en la portada principal el listado de ejemplos demostrativos y las opciones de descarga y visualización de la documentación de la API de funciones.

FIGURA 24

Captura de la pagina web para descargar y consultar sobre las funciones API del Web Framework for CGI Developers.



4.4.4 LISTADO DE FUNCIONALIDADES

Se tiene dos grupos de funciones API:

Funciones y Filtros Comunes	Filtros Basados en Convolución
• mlActiveWebCam • mlBinarize • mlBigerPixels • mlDisplay • mlDomainHSL • mlEdgeRobert • mlFilterHSV • mlGrayScale • mlHistogram • mlLoadInCanvas • mlSephia	• mlConvEmboss • mlConvGaussBlur • mlConvSharpen • mlConvSmoth • mlConvMeanRemov • mlConvEdgeSobel • mlConvolutionByUser

Para mostrar la funcionalidad de las funciones se han elaborado 6 ejemplos demostrativos:

a) Ejemplo 1 - Efectos de Edición de Fotos

Se muestra cómo usar correctamente las funciones API (Escala de grises, sepia, Binarización, detección de bordes y pixelado) y los efectos logrados mostrados sobre la imagen fuente.

Primero mostraremos el código fuente usado para definir el acabado HTML, seguidamente la captura de imagen ejecutando las funciones API.

Código HTML base, en las líneas 4 y 5 importantes para incluir el uso las funciones API, en las línea 12 la etiqueta <img id="source"> y finalmente en las líneas 20 y 21 la invocación a una función API pasando como argumento el ID.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  <script src="../JS/mlCGI.e.js" type=""></script>
5  <script src="../JS/mlDialogs.js" type=""></script>
6  <link href="../CSS/mlstyle.css" type="text/css">
7  </head>
8
9  <body>
10 <div id="mlBody" class="SMetal">
11 <section class="mlBox">
12       <img id="source" src="" width=570 height=370>
13 </section>
14 <section class="mlBoxHs">
15       <div id="h_rgba" class="cvHistog"></div>
16       <div id="h_gray" class="cvHistog"></div>
17 </section>
18
19 <section class="mlBox">
20 <button onClick="mlSephia(source)"> SEPHIA </button>
21 <button onClick="mlBinarize(source)"> BINARIZE </button>
22 </section>
23 </div>
24 </body>
25
```

FIGURA 25

Ejemplo 1, efectos fotográficos e histograma sobre la etiqueta <IMG>

b) Uso de <Canvas> e imágenes en tamaño real

Por implementación en control <CANVAS> será el único que no usara el escalado o ajuste visual de la imagen, mantendremos su dimensión real, por lo que se recomienda usar para la entrada elementos <IMG> o <VIDEO>, dejando el control Canvas de forma optativa pero no recomendada.

FIGURA 26

Ejemplo 2, uso del control <Canvas> mostrando la imagen en su tamaño real con el efecto de Binarización como resultado.

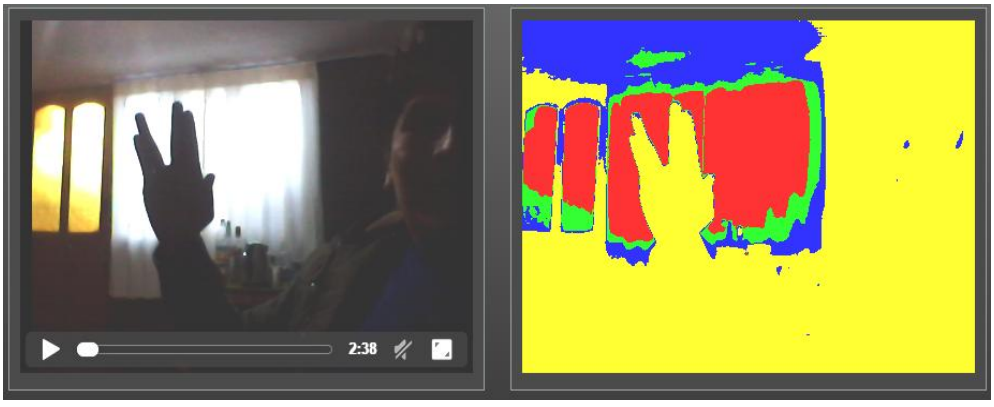
c) Procesamiento de Imágenes de Video y WebCam

El Framework soporta la captura de cuadros de video (Frames) por medio del nuevo elemento de HTML5 el control <VIDEO>, también se ha agregado una función que permite conectar la WebCam y poder procesar la entrada visual.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  <script src="../JS/mlCGI.e.js" type=""></script>
5  <link href="../CSS/mlstyle.css" type="text/css">
6  </head>
7  <body>
8  <div id="mlBody" class="SMetal">
9  <section class="mlBox">
10       <video id="source" width=422 height=300 loop>
11       <source src="../Resources/cat.webm">
12       </video>
13 </section>
14 <section class="mlBox">
15       <canvas id="result" width=422 height=300>
16             Su navegador no soporta el elemento canvas
17       </canvas>
18 </section>
19 <section class="mlBox">
20 <button onClick="mlActiveWebCam(source)">
21          ACTIVA WEBCAM</button>
22 <button onClick="mlGrayScale(source,result)">
23          BINARIZAR</button>
24 </section>
25 </div>
26 </body>
```

FIGURA 27

Ejemplo 3, procesamiento de video y entrada por WebCam.

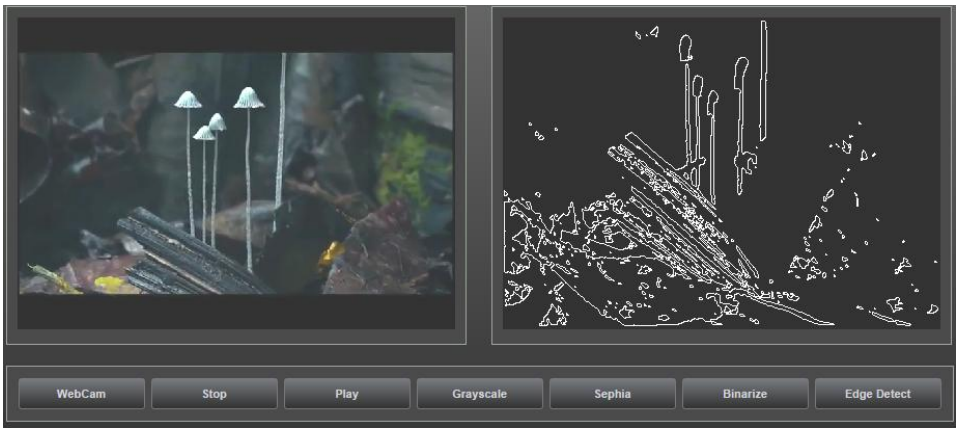


d) Procesamiento Continuo de Video

Con una sencilla rutina de activación de temporizador y selección por casos (switch), se logra procesar los fotogramas de cualquier video o entrada de WebCam.

```
1  var tmrLoop, nEffect;
2
3  function doPlayRT(){
4
5        switch( nEffect ){
6        case 1: mlGrayScale(source,result); break;
7        case 2: mlSephia(source,result);    break;
8        case 3: mlBinarize(source,result);  break;
9        case 4: doTwoFilters();             break;
10       }
11 }
12 function doTwoFilters(){
13       var canv = wcgi.DynamCanvas(source);
14       mlBinarize( canv );
15       mlEdgeRobert( canv );
16       mlDisplay( canv, result );
17 }
18 function doStopAll(){
19       window.clearInterval(tmrLoop);
20       source.pause();
21 }
22 function doInitAll(){
23       nEffect = 4;
24       source.play();
25       tmrLoop = setInterval( doPlayRT, 333 );
26 }
27 doInitAll();
```

FIGURA 28
Ejemplo 4, procesando video con temporizador y mostrando el resultado sobre un control <Canvas>



e) Histogramas de Color

Una vez procesada una imagen es posible obtener el histograma de color, la API solo requiere como argumento el identificador de una etiqueta <DIV>

```
<section class="mlBox">
      <div id="red"   class="cvHistogram"></div>
      <div id="green" class="cvHistogram"></div>
      <div id="blue"  class="cvHistogram"></div>
</section>

<script>
function doConvolutionFilters( idButton ){
switch( idButton ){
      case 2: mlConvGaussBlur(source,result); break;
      case 4: mlConvSharpen(source,result);   break;
      case 6: mlConvMeanRemov(source,result); break;
}
mlHistogram( result, red,  ML_RED );
mlHistogram( result, green,ML_RGB );
mlHistogram( result, blue, ML_GRAY );
}
</script>
```

FIGURA 29

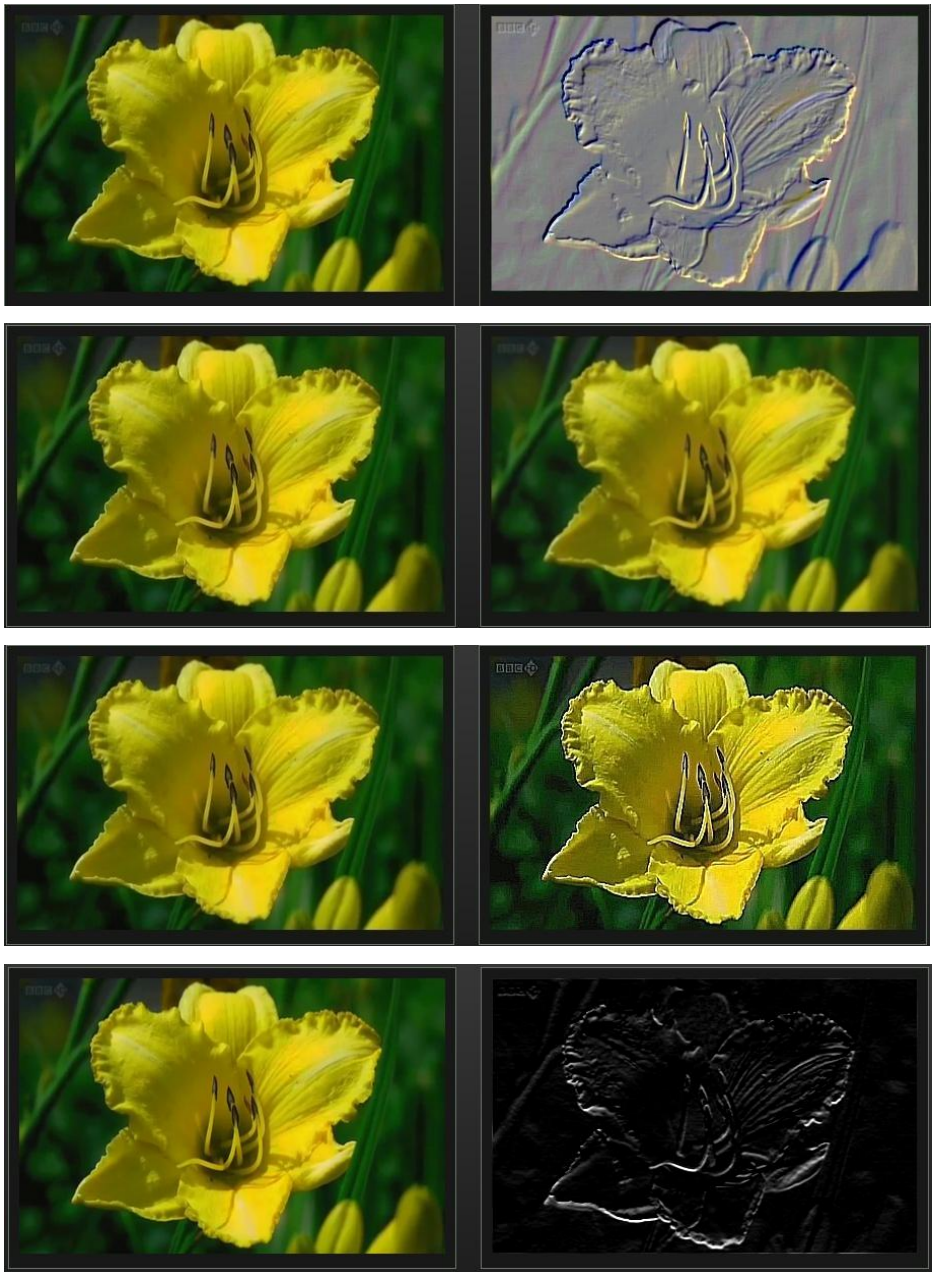
Imagen origen y resultante, desplegando tres tipos de histogramas que puede desplegar la API, de izquierda a derecha. Histograma de tonalidad roja, segundo los histogramas RGB superpuestos y tercero el histograma de grises calculado desde el histograma RGB.

f) Filtros por Convolución

Las formulas y listado de matrices de convolución listadas en el item 4.3.13, aquí se muestran con los resultados sobre la imagen origen:

FIGURA 30

primero el resultado de realce (emboss), segundo el suavizado (smoth), tercero el filtro de definición (sharpness) y cuarto el filtro para detección de bordes de sobel con barrido horizontal y vertical.



4.4.5 ANÁLISIS DEL COSTO COMPUTACIONAL

La idea de la eficiencia de los algoritmos se basa en la premisa de que el éxito de un algoritmo no debe de depender en ningún caso de la velocidad ni del potencial del sistema en que se ejecute. Un algoritmo eficiente siempre tiene que ser mejor que otro que no lo es, aun en el caso de que el segundo se ejecute en un sistema claramente superior.

Un buen algoritmo no debe depender del sistema en el que se ejecuta. Cuando hablamos del sistema informático en el entorno de análisis del coste computacional (o de otros recursos), englobamos en la expresión cualquier característica peculiar: sistema operativo, hardware, lenguaje de programación, RAM, velocidad de CPU, etc.

Coste Computacional de las funciones de efectos generales

Se tiene un bucle **for** que recorrerá la longitud dada por el producto del ancho por altura de la imagen de entrada.

```
for( var i=0; i<n; i=i+4 ){   (1) + (n/4)
pixmap[i+0] = r1;             (1)
pixmap[i+1] = r1;             (1)
pixmap[i+2] = r1;             (1)
}
```

$$f(n) = 1 + n/4 + 1 + 1 + 1$$
$$f(n) = n + 4$$

Coste computacional de las funciones de Convolución

Para el caso de la función de convolución se ha anidado 3 bucles los dos primeros para recorrer el mapa de bits especificado por la anchura x altura, agregando una iteración de 3 tiempos para recorrer los 3 tonos RGB del mapa de bits.

```
for( var i=0; i<n; i++ ){   (1) * (n)
for( var j=0; j<m; j++){      (1) * (m)
      subPix3x3 = [ getPix3x3 ]        (9)
for( k=0; k<3; k++ ){          1*(k)
res = subPix3x3 * ker3x3      (9)
      }
   }
}
```

$$f(n) = 1 * n + 1(m + 9 + (1 * k + 9))$$
$$f(n) = n * m * k + 18$$

CONCLUSIONES

Se ha implementado cuatro clases: mlMatrix, mlConvMatrix, mlChroma, mlWCGI que interactúan entre ellas para administrar, convertir los espacios de color RGB, HSV, HSL dejando un listado de funciones API, descrita en los resultados, además de los ejemplos demostrativo con una sencillez en su uso dejando al programador pocas líneas para obtener resultados óptimos. La entrada de datos es multiple desde imágenes normales, fotografías en diversos formatos gráficos como imágenes obtenidas desde video y agregando el soporte para el uso de WebCam como dispositivo de entrada de imágenes, con solo unas líneas en su aplicación.

Se ha integrado una vez adaptados los algoritmos en su mayoría escritos en C++ y algunos en C# hacia el lenguaje Javascript y su interacción con los elementos de HTML5, agregando las funcionalidades de Ajax DOM para lograr una WebApp funcional ejecutable desde cualquier navegador en Smarphones, Tablets y Laptop, que no requiera recargar el contenido sino ejecutarlo directamente sobre los controles <CANVAS>, <IMG>, <VIDEO> y <DIV>, se ha cumplido con optimizar los algoritmos en menor cantidad de líneas de código

sumando 700 líneas todo el Framework, con la gran cantidad de funcionalidades vistas en el capítulo de resultados.

Se ha documentado en formato HTML la colección de funciones API del Framework, así como los ejemplos demostrativos de cada una de las funcionalidades codificadas y adaptadas para su funcionamiento en entornos Web y navegadores que soporten el estándar HTML5, publicándose en el sitio web descrito en los anexos, disponible para la descarga y uso de nuevos programadores que deseen usarlos en proyectos de procesamiento de imágenes y patrones, teniendo en cuenta que al finalizar obtendrán una aplicación web funcional.

RECOMENDACIONES

Se recomienda el uso de este framework para simplificar la aplicación que tenga como tarea procesar imágenes y/o videos, el framework ha sido estructurado para pueda ser incrementado, modificado y por supuesto mejorado.

Puede mejorarse la funcionalidad agregando mas filtros de procesamiento, los que tendrán que ser reescritos para el lenguaje javascript o en el caso más del lado del servidor puede escribirse en lenguaje PHP, pero eso dependerá de la pericia del programador.

BIBLIOGRAFÍA

Asqui V., Richard (2009). ***Sistema de Reconocimiento de Firmas y Rubricas Digitales***. Tesis de Maestría en Informática, Universidad Nacional del Altiplano.

Bahamón C., Nelson (2011). ***Restauración de Imágenes Mediante un Modelo Matemático Basado en las Técnicas de Detección de Bordes y Propagación de Texturas***. Magister en Ciencias, Bogotá. Universidad Nacional de Colombia.

Canny, Jhon (1986). ***A Computational Approach to Edge Detection***. IEEE Member, Transactions on Pattern Analysis and Machine Intelligence, Vol PAMI-8. 6th Nov.

Carranza A., Fredy.&Florian C. Elizabeth (2008). ***Extracción De Características Para La Recuperación De Imágenes Médicas Por Contenido Utilizando Las Wavelets De Gabor***. Universidad Nacional de Trujillo. Ingeniería de Sistemas e Informática. Trujillo, Perú.

Daniel S., Leonardo (2012). **Algoritmo de Segmentación Online de Imágenes en Secuencia de Video**. Tesis de Ingeniería Informática, Universidad de Buenos Aires, Mayo de 2012.

Marcotte, Ethan (2012), ***Responsive Web Design***, (Web eBooks) A Book Apart Comunity, http://www.abookapart.com/products/responsive-web-design.

Gauchat J., Dennis (2012). **HTML5 for Masterminds: How to Take Advantages to create Amazing WebSites**. 2nd Edition. MinkBooks.

González, Rafael & Woods, Richard (1998). ***Tratamiento Digital de Imágenes***. Addison-Wesley 2da Edición, Computación Gráfica. Visión Computacional. Abril 4, 1998.

González, Rafael &Woods, Richard (2008). ***Digital Image Processing***.(Third Edition).2008 by Pearson Education, Inc. Pearson Prentice Hall.Upper Saddle River, New Jersey 07458

Shapiro G., Linda, George C. Stockman (2001),***Computer Vision*.** 1 Ed. Prentice Hall, 2001.Paperback 608 pages.

Juan D., Gauchat (2012), ***El Gran Libro de HTML5, CSS3 y Javascript***, Primera Edición, Marcombo S.A. 2012, Gran Via de les Corts Catalanes, Barcelona, España.

Bastos, Igor (2014). ***Gesture Recognition Using Shape Characteristics***. Math Institute, Federal University of Bahia, Brazil. WIP SIBGRAPI.

Otavio A.,Bruno& Valle, Eduardo (2013), ***Image and Video Representations based on Visual Dictionaries***, RECOD Lab; Institute of Computing, Universidade de Campiñas (UNICAMP), Campiñas, SP - Brazil.

Sardi L., Daniel (2012), ***Algoritmo de Segmentación Online de Imágenes en Secuencias de Video***. Universidad de Buenos Aires, Facultad de Ingeniería, Mayo 2012.

Suarez B., Alvaro (2009), ***Análisis de Métodos de Procesamiento de Imágenes Estereoscópicas Forestales***, Tesis de Maestría, Universidad Complutense de Madrid, España 2009.

Vadivel A., Sural Shamik, &A. K. Majumdar (2012).***Color-Texture Feature Extraction Using Soft Decision From HSV Color Space***. CiteSeerX.Information Sciences and Technology, Pennsylvania State University.

De Andrade, Victor&Marroquim, Ricardo. (2013), ***A Shader Library for OpenGL 4 and GLSL 4.3 Learning and Development***, Polytechnic School - UFRJ, Rio de Janeiro, Brazil, SIBGRAPHI 2013, Arequipa.

Zhong Liu, Weihai Chen (2012). ***Regions of Interest Extraction Based on HSV Color Space***.Industrial Informatics (INDIN), 10th IEEE International Conference.

ANEXOS

ANEXO A - PRE-PROCESAMIENTO

Algoritmos de Procesamiento de Imágenes

a) **Binarización:** La Segmentación por Umbral (Thresholding) es utilizada por los métodos de sustracción de fondo.

$$\begin{cases} si\ \ Pxy > Umbral \ \rightarrow PixelBlanco \\ si\ \ Pxy < Umbral \rightarrow PixelNegro \end{cases} / Pxy \in I(x,y)$$

b) **Detección de Bordes:** tiene como objetivo la identificación de puntos en una imagen, donde el brillo de la imagen cambia drásticamente o más formalmente, tiene discontinuidades.

$$\mathbf{G_x} = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G_y} = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

En cada punto de la imagen, se realiza una convolución de matrices, otra técnica para la detección de bordes es por distancias Euclidianas y se define como:

$$\mathbf{G} = \sqrt{\mathbf{G_x}^2 + \mathbf{G_y}^2}$$

c) **Convolución:** convolución es un operador matemático que transforma dos funciones **f** y **g** en una tercera función. representa la magnitud en la que se superponen **f** y una versión trasladada e invertida de **g**. Una convolución es un tipo muy general de media móvil.

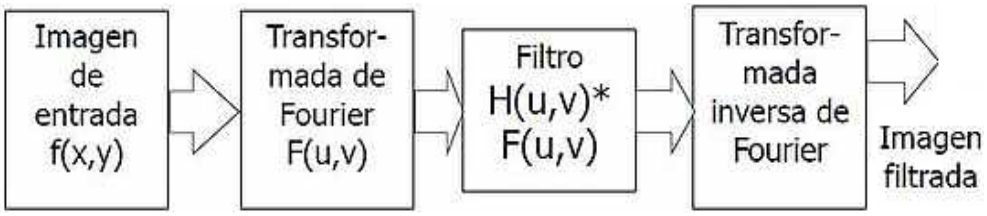
$$F(x,y) = f(x,y) * g(x,y) = \sum_i \sum_j f(x+i, y+j) h(i,j)$$

$$\begin{bmatrix} y[0] \\ y[1] \\ y[2] \\ . \\ . \\ . \\ y[2*n-1] \end{bmatrix} = \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ . \\ . \\ . \\ x[n] \end{bmatrix}^T \begin{bmatrix} h[0] & h[1] & h[2] & ... & h[n] & 0_k & ... & 0_{2*n-1} & & \\ 0 & h[0] & h[1] & h[2] & ... & h[n] & 0_k & ... & 0_{2*n-2} & \\ 0 & 0 & h[0] & h[1] & h[2] & ... & h[n] & 0_k & ... & 0_{2*n-3} \\ . & & & & & & & & & \\ . & & & & & & & & & \\ . & & & & & & & & & \\ 0 & 0 & ... & h[0] & h[1] & h[2] & ... & h[n] & & \end{bmatrix}$$

d) **Filtros:** Es el conjunto de técnicas englobadas dentro del procesamiento de imágenes cuyo objetivo es obtener, a partir de una imagen, otra cuyo resultado sea más adecuado para una aplicación específica, mejorando características que posibilite efectuar operaciones del procesado sobre ella.

FIGURA 1

(Arriba) representación del proceso de aplicación de filtros. (abajo) Imagen original, imagen con filtro de Gauss y filtro Laplaciano.



ANEXO B: IMPLEMENTACION DEMO DE CLASE *mlCGI* (beta) EN LENGUAJE JAVASCRIPT

```
/************************************************************
 *
*  Project : Cross-Pattform CGI Framework   [.b]
*  Author  : Ramiro Pedro Laura Murillo
 *            - Master Candidate - UNAP
 *            - Master's Thesis
 *
*  Date    : 11-feb-2014
 *
*  Master Informatics & Computer Science Doctorade Program
*   EPG - Universidad Nacional del Altiplano - Puno 2014
*
 ************************************************************
 * History Log
 * 12-dec-2013 : Project begins with an rusty example & demo
 * 11-feb-2014 : Base Class implementation and analysis
 *                                - invoke privSephia() as private
 *                                - invoke this.privSe() when is public
 * 12-feb-2014 : Optimization and reduce code lines
 ************************************************************/

function mlCGIx1( className ){

        var mClassName;
        var mCanvasSrc;

        this.mClassName = className;

        const ML_GRAY   =1
        const ML_SEPHIA =2
        const ML_INVERT =3
        const ML_ROBERT =4
        //-----------------------------------------------------
        // Private functions
        //-----------------------------------------------------
        function privAlloc( nRows, nCols )
        {
        var mtxTemp =new gAllocMatrix( nRows, nCols);
        }

        function getImgWidth( ImgCtrl )
        {
                var res =0;
                var img =new Image();
                img.src = ImgCtrl.src;
                return img.width;
        }

        // get the real height
        function getImgHeight( ImgCtrl )
        {
                var img =new Image();
                img.src = ImgCtrl.src;

                return img.height;
        }
```

```
		// Public functions
		//-----------------------------------------------------
		this.makeBackup =function( idImg )
		{
			var img  = document.getElementById( idImg );
			var canv = document.createElement("Canvas");
			var cntx = canv.getContext("2d");

			canv.width = img.width;
			canv.height = img.height;
			cntx.drawImage(img,0,0);

			this.mCanvasSrc = canv;
		}

		this.makeRestore =function( idImg )
		{
			var img  = document.getElementById( idImg );
			var canv =this.mCanvasSrc;
			img.src =canv.toDataURL();
		}
		//-----------------------------------------------------------
		this.doImgGrayScale =function( idImg )
		{
			var img  = document.getElementById( idImg );
			var canv = document.createElement("CANVAS");
			var cntx = canv.getContext("2d");

			canv.width =getImgWidth(img);
			canv.height =getImgHeight(img);
			cntx.drawImage(img,0,0);

			privCanvasOperation( canv, ML_GRAY );

			img.src =canv.toDataURL();
		}
	//-----------------------------------------------------------
		this.doCanvasGrayScale =function( idCanvas )
		{
			var canv = document.getElementById(idCanvas);
			privCanvasOperation( canv, ML_GRAY );
		}

		this.doImgSephia =function( idImg )
		{
			var img  = document.getElementById( idImg );
			var canv = document.createElement("CANVAS");
			var cntx = canv.getContext("2d");

			canv.width =getImgWidth(img);
			canv.height =getImgHeight(img);
			cntx.drawImage(img,0,0);
			privCanvasOperation( canv, ML_SEPHIA );
			img.src =canv.toDataURL();
		}

		this.doCanvasSephia =function( idCanvas )
		{
			var canv = document.getElementById(idCanvas);
			privCanvasOperation( canv, ML_SEPHIA );
		}

		//-----------------------------------------------------------
```

```
		this.doCanvasInvert =function( idCanvas )
		{
			var canv = document.getElementById(idCanvas);
			privCanvasOperation( canv, ML_INVERT );
		}

		this.doEdgeRoberts =function( idCanvas )
		{
			var canv = document.getElementById(idCanvas);
			privCanvasOperation( canv, ML_ROBERT );
		}

		function privCanvasOperation( Canvas, nIdOperation )
		{
			var cntx = Canvas.getContext("2d");
			var dataImage = cntx.getImageData(0,0,
					Canvas.width,Canvas.height );
			var pixmap = dataImage.data;
			var width = dataImage.width;

			switch( nIdOperation )
			{
			case ML_GRAY    : privGrayScale( pixmap );break;
			case ML_SEPHIA : privSephia( pixmap );break;
			case ML_INVERT : privInvert( pixmap );break;
			case ML_ROBERT : privEdgeRoberts( pixmap, width );
			default:break;
			}

			cntx.putImageData( dataImage,0,0);
		}

		//-------------------------------------------------------------
		// in : as a raw pixel map RGBA
		//  x : no return, work as a pointer from source
		//-------------------------------------------------------------
		function privSephia( pixmap )
		{
			for( var i=0; i<pixmap.length; i+=4)// RGBA
			{
				var mean =(pixmap[i]+pixmap[i+1]+ pixmap[i+2])/3;
				pixmap[i+0]= mean*0.92;// red on top
				pixmap[i+1]= mean*0.75;// green a litle down
				pixmap[i+2]= mean*0.65;// blue mean
			}
		}

		function privGrayScale( pixmap )
		{
			for( var i=0; i<pixmap.length; i+=4)// RGBA
			{
			var mean =(pixmap[i]+ pixmap[i+1]+ pixmap[i+2])/3;
				pixmap[i+0]= mean;
				pixmap[i+1]= mean;
				pixmap[i+2]= mean;
			}
		}
	}

	//-- EOF
```

ANEXO C: IMPLEMENTACION FINAL DE LA CLASE mlWCGI

```
/*************************************************************
 *
*  Project : Cross-Pattform CGI Framework   [ver .e]
*  Author  : Ramiro Pedro Laura Murillo
 *            - Master Candidate - UNAP
 *            - Master's Thesis
 *
*  Master Informatics & Computer Science Doctorade Program
*   EPG - Universidad Nacional del Altiplano - Puno 2014
*
 *************************************************************
 * History Log
 *
 * 03-ene-2015 : Add Dynamic File Dialog
 * 03-ene-2015 : Convolution debugging and optimizing
 * 02-ene-2015 : Convolution routines
 *
 * 30-dic-2014 : Increase new iteration and Internet update
 * 29-dic-2014 : Improve WebCam Interface on MultiBrowsers
 * 29-dic-2014 : RealTime Edge Detection and WebPublishing
 * 27-dic-2014 : Croma class as color convers admin
 * 27-dic-2014 : Add some fix code lines
 * 26-dic-2014 : Correct Zoom Treatment
 * 14-dic-2014 : Media Fussion WEbCam + Video + Image
 *
 * 12-feb-2014 : Optimization and reduce code lines
 * 11-feb-2014 : Base Class implementation and analysis
 *                          - invoke privSephia() as private
 *                          - invoke this.privSe() when is public
 * 12-dec-2013 : Project begins with an rusty example & demo
 *************************************************************/

constML_DEV_NULL   = 0
constML_DEV_VIDEO  = 1
constML_DEV_IMAGE  = 2
constML_DEV_CANVAS = 3

constML_RED    = 0
constML_GREEN  = 1
constML_BLUE   = 2
constML_GRAY   = 3
constML_RGBA   = 4

constML_SEPHIA = 6
constML_INVERT = 5
constML_ROBERT = 7
//---------------------------------------------------------------------------
// Object Instance
//---------------------------------------------------------------------------
varwcgi = new mlWCGI("t1");
varwrgb = new mlChroma("t1");

//---------------------------------------------------------------------------
// Class : Convolution Matrix Definition
//---------------------------------------------------------------------------
functionmlConvMatrix( ) {
       this.factor = 1;
       this.offset = 3;
       this.M = [
              [1, 1, 1],
              [1, 1, 1],
              [1, 1, 1],
       ];
}
```

```
//--------------------------------------------------------------------------
// Class : Matrix Definition
//--------------------------------------------------------------------------
functionmlMatrix( nRows, mCols ) {
this.length = nRows;
for(var i=0; i<nRows; i++)
this[i] = new Array(mCols);
returnthis;
}


//--------------------------------------------------------------------------
// Class : Chroma Handling
//--------------------------------------------------------------------------
functionmlChroma( className ) {

	this.mClassName = className;

	this.toStrRGB = function ( r, g, b ){
		return"RGB( " +r+ ", " +g+ ", " +b+ " )";
	}

	this.hsvToRgb = function(hue, saturation, value ){

	varh = parseInt(hue * 6); // 360*
	varf = hue * 6 - h;       // dif
	varp = value * ( 1 - saturation );
	varq = value * ( 1 - saturation * f );
	vart = value * ( 1 - saturation - (1 - f) );

	switch(h) {
	case0: return this.toArrRGB( value, t, p );
	case1: return this.toArrRGB( q, value, p );
	case2: return this.toArrRGB( p, value, t );
	case3: return this.toArrRGB( p, q, value );
	case4: return this.toArrRGB( t, p, value );
	case5: return this.toArrRGB( value, p, q );
	    }
	}

	this.toArrRGB = function( r, g, b){
		return[ Math.abs( parseInt(r * 256) ),
				Math.abs(parseInt(g * 256) ),
				Math.abs(parseInt(b * 256) ) ];
	}
}


//--------------------------------------------------------------------------
// Class : CGI Handling definition Class
//--------------------------------------------------------------------------
functionmlWCGI( className ) {

	this.mClassName = className;

	this.GetDevice = function ( idDev)
	{
		if(idDev == "[object HTMLVideoElement]" )
			returnML_DEV_VIDEO;
		else if( idDev == "[object HTMLImageElement]" )
			returnML_DEV_IMAGE;
		else if( idDev == "[object HTMLCanvasElement]" )
			returnML_DEV_CANVAS;
		else{
			alert("ClassError: Object Undefined" );
			returnML_DEV_NULL;
		}
```

```
}

this.NewCanvas = function(width, height ){
	varcanv = document.createElement("Canvas");
	canv.width = width;
	canv.height = height;
	returncanv;
}

this.DynamCanvas = function(media )
{
	varcanv = document.createElement("Canvas");
	this.DrawInCanvas(canv, media );
	returncanv;
}

this.GetContext = function(idCanvas )
{
	returnidCanvas.getContext("2d");
}

this.DrawInCanvas = function(idCanvas, idSource )
{
	idCanvas.width = idSource.width;
	idCanvas.height = idSource.height;
	varctx = this.GetContext( idCanvas ); // idSource.clientWidth
	ctx.drawImage(idSource, 0, 0, idCanvas.width, idCanvas.height );
	//ctx.drawImage( idSource, 0, 0 );  // 100%
}

this.GetDataImage = function(idCanvas )
{
	varcntx = this.GetContext( idCanvas );
	returncntx.getImageData(0,0,idCanvas.width,idCanvas.height);
}

this.privSephia = function(idCanvas )
{
	varcntx   = this.GetContext( idCanvas );
	vardataIm = this.GetDataImage( idCanvas );
	varpixmap = dataIm.data;

	for(var i=0; i<pixmap.length; i+=4 ) // RGBA
	{
		varmean = (pixmap[i] + pixmap[i+1] + pixmap[i+2]) / 3;
		pixmap[i+0] = mean * 0.92;  // red on top
		pixmap[i+1] = mean * 0.75;  // green a litle down
		pixmap[i+2] = mean * 0.65;  // blue mean
	}
	cntx.putImageData(dataIm, 0, 0 );
}

this.privGrayScale = function(idCanvas )
{
	varcntx   = this.GetContext( idCanvas );
	vardataIm = this.GetDataImage( idCanvas );
	varpixmap = dataIm.data;

	for(var i=0; i<pixmap.length; i+=4 ) // RGBA
	{
		varmean = (pixmap[i] + pixmap[i+1] + pixmap[i+2]) / 3;
		pixmap[i+0] =
		pixmap[i+1] =
		pixmap[i+2] = mean;
	}
	cntx.putImageData(dataIm, 0, 0 );
}
```

```
this.getPixelPos = function(idCanvas, x, y )
{
	returnparseInt( (y * idCanvas.width + x) * 4 );
}

this.privEdgeRobert = function(idCanvas, Threshold )
{
	varcntx    = this.GetContext( idCanvas );
	vardataIm = this.GetDataImage( idCanvas );
	varpixmap = dataIm.data;

	// Correct Process - Binarize - Edge Detect
	for(y = 0; y <idCanvas.height; y++) {
		for(x = 0; x <idCanvas.width; x++) {

		p1 = this.getPixelPos(idCanvas, x, y );
		p2 = this.getPixelPos(idCanvas, x+1, y );

		cR= Math.pow( Math.abs( pixmap[p1+0]-pixmap[p2+0] ), 2 );
		cG= Math.pow( Math.abs( pixmap[p1+1]-pixmap[p2+1] ), 2 );
		cB= Math.pow( Math.abs( pixmap[p1+2]-pixmap[p2+2] ), 2 );
		nPix1  =Math.sqrt(cR + cG + cB);

		p2 = this.getPixelPos(idCanvas, x+1, y+1 );
		cR= Math.pow( Math.abs( pixmap[p1+0]-pixmap[p2+0] ), 2 );
		cG= Math.pow( Math.abs( pixmap[p1+1]-pixmap[p2+1] ), 2 );
		cB= Math.pow( Math.abs( pixmap[p1+2]-pixmap[p2+2] ), 2 );
		nPix2  =Math.sqrt(cR + cG + cB);

		nPixel= ( nPix1>=Threshold || nPix2>=Threshold )? 255:0;

		pixmap[p1+0] =
		pixmap[p1+1] =
		pixmap[p1+2] = nPixel;
	}	}

	cntx.putImageData(dataIm, 0, 0 );
}


this.privDomainHSL = function(idCanvas )
{
	varcntx    = this.GetContext( idCanvas );
	vardataIm = this.GetDataImage( idCanvas );
	varpixmap = dataIm.data;
	varTL = 512;

	for(var i=0; i<pixmap.length; i+=4 ) // RGBA
	{
		varsum = pixmap[i] + pixmap[i+1] + pixmap[i+2];

		if(sum >TL)  color = [255, 0, 0];
		else if(sum >TL * 0.6)  color = [0, 255, 0];
		else if(sum >TL * 0.25) color = [0, 0, 255];
		elsecolor = [252, 255, 0];

		pixmap[i+0] = color[0];
		pixmap[i+1] = color[1];
		pixmap[i+2] = color[2];
	}
	cntx.putImageData(dataIm, 0, 0 );
}



//-----------------------------------------------------
// Convolution Implementation area
//-----------------------------------------------------
```

```
this.getPix = function(pixmap, width, x, y )
{
	varoffset = (y * width + x) * 4;

	return[ pixmap[offset+0],   // R
		pixmap[offset+1],   // G
		pixmap[offset+2] ]; // B
		pixmap[offset+3] ]; // A
}

this.doConvolution3x3 = function(idCanvas, convKer )
{
	// mem and clean copy
	vardta = this.GetDataImage( idCanvas );
	varpix = dta.data;
	varw   = idCanvas.width;

	varcntx   = this.GetContext( idCanvas );
	vardataIm = this.GetDataImage( idCanvas );
	varpixmap = dataIm.data;

	// matrix sweep except borders
	for(y = 1; y <idCanvas.height-1; y++) {
		for(x = 1; x <idCanvas.width-1; x++) {
			//
			// [ (x-1,y-1), (x,y-1), (x+1,y-1) ],
			// [ (x-1,  y), ( x, y), (x+1, y ) ],
			// [ (x-1,y+1), (x,y+1), (x+1,y+1) ],
			//
			pixRGBA= [
				[ this.getPix(pix,w,x-1,y-1),
				this.getPix(pix,w,x,y-1),
				this.getPix(pix,w,x+1,y-1) ],
				[ this.getPix(pix,w,x-1,y  ),
				this.getPix(pix,w,x,y  ),
				this.getPix(pix,w,x+1,y  ) ],
				[ this.getPix(pix,w,x-1,y+1),
				this.getPix(pix,w,x,y+1),
				this.getPix(pix,w,x+1,y+1) ],
			];

			offset= this.getPixelPos( idCanvas, x, y );
			for(i=ML_RED; i<=ML_BLUE; i++ ){
			pixRes= pixRGBA[0][0][i] * convKer.M[0][0] +
				pixRGBA[0][1][i] * convKer.M[0][1] +
				pixRGBA[0][2][i] * convKer.M[0][2] +
				pixRGBA[1][0][i] * convKer.M[1][0] +
				pixRGBA[1][1][i] * convKer.M[1][1] +
				pixRGBA[1][2][i] * convKer.M[1][2] +
				pixRGBA[2][0][i] * convKer.M[2][0] +
				pixRGBA[2][1][i] * convKer.M[2][1] +
				pixRGBA[2][2][i] * convKer.M[2][2] ;

			pixRes= ( pixRes / convKer.factor ) + convKer.offset ;

				if(pixRes <0)   pixRes = 0;
				if(pixRes >255) pixRes = 255;

				pixmap[offset+i ] = pixRes;
			}
		}
	}
	cntx.putImageData(dataIm, 0, 0 );
}
}
```

```
//-------------------------------------------------------------------------------
// API : export functions
//-------------------------------------------------------------------------------
functionmlLoadInCanvas( idCanvas, srcFile ){
	varimg = new Image();
	img.src = srcFile;
	img.onload = function(){
		wcgi.DrawInCanvas(idCanvas, img );
	}
}
//-------------------------------------------------------------------------------
functionmlDisplay( idSource, idDestin ){
	// all media draws it over canvas
	if(wcgi.GetDevice(idDestin) == ML_DEV_CANVAS ){
		wcgi.DrawInCanvas(idDestin, idSource );
	}
	// conversion media over img
	else if( wcgi.GetDevice(idDestin) == ML_DEV_IMAGE ){

		if(wcgi.GetDevice(idSource) == ML_DEV_IMAGE ){
			idDestin.src = idSource.src;
		}

		if(wcgi.GetDevice(idSource) == ML_DEV_VIDEO ){
			varcanv = wcgi.DynamCanvas(idSource);
			idDestin.src = canv.toDataURL();
		}

		if(wcgi.GetDevice(idSource) == ML_DEV_CANVAS ){
			idDestin.src = idSource.toDataURL();
		}
	}
	else{
		alert("Can't write Output");
	}
}
//-------------------------------------------------------------------------------
functionmlActiveWebCam( idVideo ){
	wcgi.AttachWebCam(idVideo );
}
//-------------------------------------------------------------------------------
functionmlFilterHSV( idSource, idDestin ){
	varcanv = wcgi.DynamCanvas(idSource);
	h = Math.random()*0.6;
	s = Math.random()*0.50;  //s = 0.20;
	v = Math.random()*0.20;  //v = 0.40;
	wcgi.privFilterHSV(canv, h, s, v );
	mlDisplay(canv, (idDestin)?idDestin:idSource );
}
//-------------------------------------------------------------------------------
functionmlDomainHSL( idSource, idDestin ){
	varcanv = wcgi.DynamCanvas(idSource);
	wcgi.privDomainHSL(canv );
	mlDisplay(canv, (idDestin)?idDestin:idSource );
 }
//-------------------------------------------------------------------------------
functionmlGrayScale( idSource, idDestin ){
	varcanv = wcgi.DynamCanvas(idSource);
	wcgi.privGrayScale(canv );
	mlDisplay(canv, (idDestin)?idDestin:idSource );
}
//-------------------------------------------------------------------------------
functionmlSephia( idSource, idDestin ){
	varcanv = wcgi.DynamCanvas(idSource);
	wcgi.privSephia(canv );
	mlDisplay(canv, (idDestin)?idDestin:idSource );
```

```
}

//--------------------------------------------------------------------------
functionmlBinarize( idSource, idDestin ){
	varcanv = wcgi.DynamCanvas(idSource);
	wcgi.privBinarize(canv, 75 );
	mlDisplay(canv, (idDestin)?idDestin:idSource );
}
//--------------------------------------------------------------------------
functionmlEdgeRobert( idSource, idDestin ){
	varcanv = wcgi.DynamCanvas(idSource);
	wcgi.privEdgeRobert(canv, 25 );
	mlDisplay(canv, (idDestin)?idDestin:idSource );
}
//--------------------------------------------------------------------------
functionmlBigerPixels( idSource, idDestin ){
	varcanv = wcgi.DynamCanvas(idSource);
	wcgi.privBigerPixels(canv, 8 );
	mlDisplay(canv, (idDestin)?idDestin:idSource );
}
//--------------------------------------------------------------------------
functionmlConvEmboss( idSource, idDestin ){
	varkerConv = new mlConvMatrix();

	kerConv.offset = 127;
	kerConv.factor = 1;
	kerConv.M = [ [ 2,  0,  0 ],
	              [ 0, -1,  0 ],
	              [ 0,  0, -1 ], ];

	varcanv = wcgi.DynamCanvas(idSource);
	wcgi.doConvolution3x3(canv, kerConv );
	mlDisplay(canv, (idDestin)?idDestin:idSource );
}
//--------------------------------------------------------------------------
functionmlConvGaussBlur( idSource, idDestin ){
	varnWeight = 4;
	varkerConv = new mlConvMatrix();

	kerConv.offset = 3;
	kerConv.factor = nWeight + 12;
	kerConv.M = [ [ 1,    2,    1],
	              [ 2, nWeight, 2],
	              [ 1,    2,    1], ];

	varcanv = wcgi.DynamCanvas(idSource);
	wcgi.doConvolution3x3(canv, kerConv );
	mlDisplay(canv, (idDestin)?idDestin:idSource );
}
//--------------------------------------------------------------------------
functionmlConvSharpen( idSource, idDestin ){
	varnWeight = 10;
	varkerConv = new mlConvMatrix();

	kerConv.offset = 3;
	kerConv.factor = nWeight - 4;
	kerConv.M = [ [  0,    -1,    0 ],
	              [ -1, nWeight, -1 ],
	              [  0,    -1,    0], ];

	varcanv = wcgi.DynamCanvas(idSource);
	wcgi.doConvolution3x3(canv, kerConv );
	mlDisplay(canv, (idDestin)?idDestin:idSource );
}
//--------------------------------------------------------------------------
functionmlConvSmoth( idSource, idDestin ){
	varnWeight = 4;
	varkerConv = new mlConvMatrix();
```

```
        kerConv.offset = 3;
        kerConv.factor = nWeight + 8;
        kerConv.M = [ [ 1,    1,    1 ],
                      [ 1, nWeight, 1 ],
                      [ 1,    1,    1 ], ];

        varcanv = wcgi.DynamCanvas(idSource);
        wcgi.doConvolution3x3(canv, kerConv );
        mlDisplay(canv, (idDestin)?idDestin:idSource );
}
//---------------------------------------------------------------------------
functionmlConvMeanRemov( idSource, idDestin ){
        varnWeight = 9;
        varkerConv = new mlConvMatrix();

        kerConv.offset = 3;
        kerConv.factor = nWeight - 8;
        kerConv.M = [ [ -1,   -1,    -1 ],
                      [ -1, nWeight, -1 ],
                      [ -1,   -1,    -1 ], ];

        varcanv = wcgi.DynamCanvas(idSource);
        wcgi.doConvolution3x3(canv, kerConv );
        mlDisplay(canv, (idDestin)?idDestin:idSource );
}
//---------------------------------------------------------------------------
functionmlConvEdgeSobel( idSource, idDestin ){

        varkerConv = new mlConvMatrix();
        varcanv = wcgi.DynamCanvas(idSource);

        kerConv.offset = 0;
        kerConv.factor = 1;

        wcgi.privGrayScale(canv );

        kerConv.M = [ [ 1, 0, -1 ],
                      [ 2, 0, -2 ],
                      [ 1, 0, -1 ], ];
        wcgi.doConvolution3x3(canv, kerConv );

        kerConv.M = [ [  1,  2,  1 ],
                      [  0,  0,  0 ],
                      [ -1, -2, -1 ], ];

        wcgi.doConvolution3x3(canv, kerConv );
        mlDisplay(canv, (idDestin)?idDestin:idSource );
}

//- EOF
```

ANEXO D: ALGORITMO DE CONVOLUCION OPTIMIZADO

```
//-----------------------------------------------------
// Convolution Implementation area
//-----------------------------------------------------
this.getPix = function(pixmap, width, x, y )
{
	varoffset = (y * width + x) * 4;

	return[ pixmap[offset+0],    // R
		pixmap[offset+1],    // G
		pixmap[offset+2] ]; // B
		pixmap[offset+3] ]; // A
}

this.doConvolution3x3 = function(idCanvas, convKer )
{
	vardta = this.GetDataImage( idCanvas );
	varpix = dta.data;
	varw   = idCanvas.width;

	varcntx   = this.GetContext( idCanvas );
	vardataIm = this.GetDataImage( idCanvas );
	varpixmap = dataIm.data;

	// matrix sweep except borders
	for(y = 1; y <idCanvas.height-1; y++) {
		for(x = 1; x <idCanvas.width-1; x++) {
			//
			// [ (x-1,y-1), (x,y-1), (x+1,y-1) ],
			// [ (x-1,  y), ( x, y), (x+1, y ) ],
			// [ (x-1,y+1), (x,y+1), (x+1,y+1) ],
			//
			pixRGBA= [
				[ this.getPix(pix,w,x-1,y-1),
				this.getPix(pix,w,x,y-1),
				this.getPix(pix,w,x+1,y-1) ],
				[ this.getPix(pix,w,x-1,y  ),
				this.getPix(pix,w,x,y  ),
				this.getPix(pix,w,x+1,y  ) ],
				[ this.getPix(pix,w,x-1,y+1),
				this.getPix(pix,w,x,y+1),
				this.getPix(pix,w,x+1,y+1) ],
			];

			offset= this.getPixelPos( idCanvas, x, y );
			for(i=ML_RED; i<=ML_BLUE; i++ ){
			pixRes= pixRGBA[0][0][i] * convKer.M[0][0] +
				pixRGBA[0][1][i] * convKer.M[0][1] +
				pixRGBA[0][2][i] * convKer.M[0][2] +
				pixRGBA[1][0][i] * convKer.M[1][0] +
				pixRGBA[1][1][i] * convKer.M[1][1] +
				pixRGBA[1][2][i] * convKer.M[1][2] +
				pixRGBA[2][0][i] * convKer.M[2][0] +
				pixRGBA[2][1][i] * convKer.M[2][1] +
				pixRGBA[2][2][i] * convKer.M[2][2] ;

			pixRes= ( pixRes / convKer.factor ) + convKer.offset ;

				pixmap[offset+i ] = pixRes;
			}
		}
	}
	cntx.putImageData(dataIm, 0, 0 );
}
```