Universidad Nacional del Litoral

Facultad de Ingeniería y Ciencias Hídricas
Ingeniería en Informática

Proyecto Final de Carrera

Implementación de una biblioteca para el procesamiento de imágenes desarrollada con tecnologías web

Autor: Fort Villa, Alejandro
Director: Dr. Albornoz, Enrique Marcelo
Co-Director: Dr. Martínez, Cesar

27 de julio de 2015

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
A. Fort Villa, E. M. Albornoz & C. E. Martínez; "Implementación de una biblioteca para el procesamiento de imágenes desarrollada con tecnologías web"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2015.

Resumen

El procesamiento digital de imágenes (PDI) es un área que se encarga de estudiar las propiedades de las imágenes y en la cual se definen técnicas que permiten analizar y modificar las imágenes con fines estéticos o bien, para resaltar y extraer determinada información. Las herramientas con las que cuenta el campo del PDI son muy variadas dependiendo del área de aplicación en la cual se desea trabajar. Una de las herramientas más usadas por los desarrolladores e investigadores son las bibliotecas de programación. Una biblioteca es un conjunto de funciones implementadas mediante lenguaje de programación que le brindan al usuario desarrollador un grupo de elementos que ofrecen una interfaz acabada acorde a una funcionalidad específica.

El presente proyecto consiste en la implementación de una biblioteca para PDI realizada en su totalidad en JavaScript compatible íntegramente con HTML5[1], permitiendo compatibilidad con cualquier navegador de internet, independientemente del dispositivo que se utilice y sin necesidad de ningún programa o aplicación particular más que el mismo explorador. Para su implementación se utilizó un diseño estructural basado en la interfaz de la herramienta para el procesamiento de imágenes OpenCV[2], aprovechando la popularidad y la vasta documentación de la misma, facilitando la migración desde ésta. La biblioteca se definió en un paradigma de programación Orientado a Datos, que se integra con el diseño y evita la redundancia de métodos en memoria, además de ser más eficiente, ya que no utiliza intermediaciones entre los datos [15].

Para tener una visión más general del comportamiento de este desarrollo, se presentan un conjunto de pruebas que involucran un análisis comparativo que contrasta los resultados obtenidos con la biblioteca desarrollada y dos bibliotecas de referencia. Además, se incluye una evaluación de las rutinas en tiempo real, utilizando streams de una cámara web estándar.

[1] HyperText Markup Language: Lenguaje de marcas utilizado para la creación de páginas web.
[2] http://sourceforge.net/projects/opencvlibrary/

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
A. Fort Villa, E. M. Albornoz & C. E. Martínez; "Implementación de una biblioteca para el procesamiento de imágenes desarrollada con tecnologías web"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2015.

Prefacio

El ámbito web es un entorno que ha ido en creciente desarrollo en la última década. En la actualidad son cada vez más los servicios y herramientas que se brindan a través de internet, y los dispositivos que permiten la navegación a través de los sitios ya están naturalizados en la sociedad, quedando al alcance de cualquier persona. En este ámbito, el procesamiento digital de imágenes, ha crecido como área de investigación y tiene aplicación en infinidades de recursos disponibles en la web, como juegos, aplicaciones artísticas, de detección de formas, reconocimiento facial, y algunos sistemas ya naturalizados en el entorno, como los SIG (Sistema de Información Geográficos) y los sistemas de edición de imágenes.

El desarrollo de una biblioteca para el procesamiento de imágenes con herramientas web clientes, nace originalmente como una necesidad del autor, pero luego pasó a ser de interés general. Finalmente implementada, fue utilizada en varios proyectos, como aplicaciones de efectos artísticos, reconocimiento de formas en tiempo real, algoritmos de detección de daltonismo, entre otros. Si bien existen herramientas para procesamientos de imágenes en tecnologías web, la oferta particular de bibliotecas desarrolladas para uso exclusivo del lado cliente, es prácticamente nula. La ventaja principal de desarrollar sobre una tecnología alojada en el lado cliente del modelo Cliente - Servidor, es la de estar en contacto directo con el usuario, permitiendo la interacción entre el este y la aplicación, sin tener servicios intermediarios. Además se aprovecha la eficiencia computacional con la que cuenta la tecnología con la cual se accede a la herramienta.

En este trabajo se presenta una biblioteca para el procesamiento de imágenes, con una interfaz basada en la de la biblioteca OpenCV (implementada en lenguaje C), que permite editar imágenes de cualquier formato soportado por el navegador del usuario. Se implementaron métodos que incluyen manipulación de modelos de color, filtrado espacial, filtrado frecuencial y operaciones morfológicas, además de métodos de control como, manipulación de profundidad de bytes del pixel, cambio de dimensión de la imagen y representación de histogramas. El conjunto de métodos desarrollados en el presente proyecto constituyen los procesamientos elementales del PDI y permiten tener una funcionalidad básica, dejando la extensión y mejora de los mismos para trabajos futuros. El diseño de biblioteca está basado en módulos separados, integrados bajo una misma clase, representando un patrón de diseño

singleton[3]. Para la implementación se utilizó un paradigma de programación orientado a datos (POD). En lo que respecta a las tecnologías utilizadas. La biblioteca fue desarrollada íntegramente en JavaScript. JavaScript es el lenguaje más utilizado para manipular elementos dinámicos en HTML.

El informe se encuentra organizado en 5 capítulos, como se explica a continuación.

En el Capítulo 1, se expone la motivación central del desarrollo de este trabajo, seguido de una reseña del estado del arte de las tecnologías web, haciendo hincapié en las características del modelo Cliente - Servidor. Posteriormente, se describen los objetivos generales y específicos, seguidos de los alcances del presente trabajo.

En el Capítulo 2, se tratan conceptos y métodos del marco teórico del procesamiento de imágenes, y posteriormente las características técnicas de la tecnología utilizada.

El Capítulo 3 presenta el diseño estructural propuesto, así como su justificación. Posteriormente se detalla el esquema de implementación. También, se describen detalladamente las etapas del algoritmo general de uso de la biblioteca.

En el Capítulo 4, se exponen los experimentos y los resultados obtenidos mediante comparaciones con otras herramientas de referencia. También se propone una prueba del uso de la biblioteca en tiempo real. Se concluye el capítulo con un análisis de compatibilidad en diversos navegadores.

Finalmente, en el Capítulo 5, se exponen las conclusiones finales y los desarrollos futuros para corto, mediano y largo plazo.

Alejandro Fort Villa
Santa Fe, Argentina
8 de junio 2015

[3]Patrón de diseño que garantiza una única instancia de una clase o variable

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
A. Fort Villa, E. M. Albornoz & C. E. Martínez; "Implementación de una biblioteca para el procesamiento de imágenes desarrollada con tecnologías web"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2015.

Indice general

Indice de figuras

CAPITULO 1

Introducción

En este capítulo se presenta la motivación del desarrollo del proyecto, seguido de una descripción del estado del arte de la tecnología usada y algunos conceptos particulares que ayudan a comprender la motivación. Luego, se exponen los objetivos generales y específicos. Finalmente se describen los alcances del proyecto.

1.1. Motivación

En la última década se ha consolidado la tendencia que consiste en la migración de datos y servicios *standalone*[1] por sus análogos en internet (servicios en la nube, *Cloud computing*)[2]. Hoy en día, un gran número de aplicaciones e incluso Sistemas Operativos, radican íntegramente en internet [1]. Simultáneamente, las redes de comunicación inalámbricas se hicieron más comunes y nuevos protocolos brindan acceso a internet a dispositivos que anteriormente cumplían funciones más básicas. Inmediatamente se masificaron los dispositivos móviles carentes de discos de almacenamiento que basan su operatividad en el acceso a las redes móviles, afianzados en la practicidad de un entorno simple y respaldados por un sinfín de aplicaciones situadas en internet que reemplazan la potencialidad que brinda una computadora.

Un tipo de aplicaciones muy requeridas son aquellas que involucran algún tipo de procesamiento de imagen, los ejemplos más comunes son: sistemas de información geográfica (GIS, del inglés Geographic Information Systems), sistemas de posicionamiento global (GPS, del inglés Global Positioning System), aplicaciones de artes gráficas, juegos y reconocimiento automático de formas, entre otras. Una de las claves en el diseño de estas aplicaciones, y que comandan todo su desarrollo, es la

[1] Aplicaciones que se controlan y ejecutan como entidad independiente.
[2] Paradigma que se basa en ofrecer servicios a través de internet.

elección del lugar donde operarán: lado servidor o lado cliente del modelo Cliente - Servidor [2]. Las primeras trabajan independientemente del dispositivo, pero carecen de interactividad y pueden sobrecargar el procesador del servidor, son ejemplos: GD, ImageJ o PIL, todas son bibliotecas para el procesamiento de imágenes desarrolladas en C, Java y Python, respectivamente. Por otra parte, las herramientas del lado cliente permiten interactividad y operatividad en tiempo real. A pesar de las desventajas mencionadas, aquellas aplicaciones que deben implementar procesamientos de imágenes deben optar por tecnologías del lado servidor, pues son las únicas que brindan herramientas para tales operaciones.

En 2009 la W3C habilita la quinta revisión de HTML [5, 6].Uno de los puntos más fuertes de esta versión es que se incorporó un objeto que permite acceder a la información detallada de las imágenes, aunque carece de métodos de procesamiento y de una interfaz cómoda [7, 18, 20].

Este proyecto se gesta desde la necesidad de contar con una biblioteca de métodos para el procesamiento de imágenes que permita acceder a los valores de la misma y sea desarrollada con tecnologías web estándar del lado cliente. Que incorpore, además, métodos de interfaz intuitivos basados en tecnologías comúnmente utilizadas, acelerando la familiarización del usuario. Permitiendo, también, realizar algoritmos de procesamiento e incorporar los resultados en el elemento CANVAS - HTML, de manera que pueda ser visualizado desde cualquier navegador independientemente del dispositivo.

1.2. Estado del arte

Desde sus orígenes hasta hoy, la tecnología web, se ha ido complejizando considerablemente. Esto se debe, principalmente, a la mejora constante de los dispositivos visualizadores de contenido, pero sobre todo, al aumento del volumen de datos manejados y de los servicios ofrecidos mediante internet.

El contexto web es muy extenso y son muchos los actores y protocolos intervinientes, aunque es posible establecer un marco guía a partir del modelo Cliente - Servidor [2]. Este proyecto se basa en explotar las bondades que brinda la estructura cliente de dicho modelo.

1.2.1. Modelo Cliente - Servidor

La arquitectura Cliente - Servidor fue diseñada a finales de la década de 1960 como un sistema codificación y decodificación entre dos computadoras. Este modelo de comunicación es el que se utiliza hoy en día como esquema de comunicación en internet en la mayoría de los casos.

El esquema Cliente - Servidor consiste, básicamente, en establecer un inter-

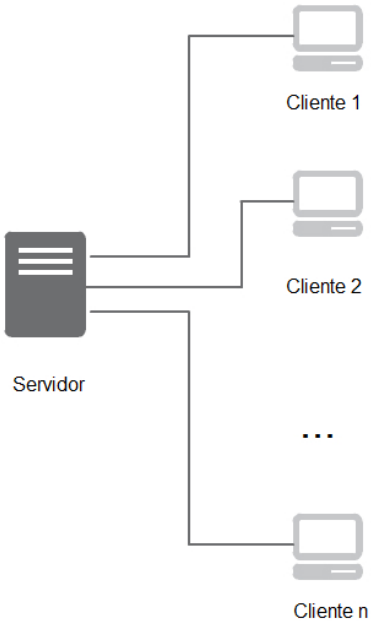


Figura 1.1: Esquema Cliente - Servidor.

cambio de tareas entre varias computadoras o terminales con el fin de que cada parte realice sólo aquellas funciones que son más adecuadas para sus condiciones, optimizando recursos y aliviando las vías de comunicación. En la Figura 1.1 se ve un esquema de n clientes interactuando con un servidor.

La relación Cliente - Servidor es una relación del tipo lógica, donde el servidor no se ejecuta necesariamente en una misma máquina ni administra una única tarea. Incluso un servidor puede brindar un servicio para un grupo de clientes y ser cliente de otro servidor. Es por esta característica que a este modelo se lo conoce como una estructura distribuida. El uso más común del modelo en internet asigna las tareas que tienen que ver con la visualización de entornos gráficos e interacción con el usuario a las terminales clientes, mientras que la administración, integridad y seguridad de los datos quedan a cargo del lado servidor [2].

En la Figura 1.2 se ve un grupo de clientes consumiendo servicios de un servidor de aplicaciones, que a su vez es cliente de un servidor de base de datos.

1.2.2. Aplicación web

Una aplicación web es un software que permite acceder a datos administrados por un servidor web a través de internet o intranet mediante un navegador web. La estructura general de una aplicación web suele involucrar 3 elementos característicos: Una interfaz gráfica desarrollada con tecnologías compatibles con el navegador

Figura 1.2: Esquema Cliente - Servidor Aplicaciones - Servidor de Base de datos.

web, como HTML, CSS[3] y algún lenguaje interpretado como JavaScript. El segundo elemento es el servidor web que permite interpretar el protocolo HTTP[4] para comunicarse con la interfaz y hacer de intermediario con el tercer elemento. El último elemento suele ser algún administrador de datos persistentes como un servidor de base de datos o un servidor de aplicaciones.

En los últimos años muchas aplicaciones han migrado de una versión dedicada a una versión web. Esto es debido a que las aplicaciones web presentan algunas ventajas que las hace propicias para el desarrollo de su función [2, 4]. Dentro de las bondades que brindan las aplicaciones web se puede mencionar que:

- Son independientes del sistema operativo instalado en el dispositivo.
- No poseen redundancia de recursos ante múltiples clientes, ya que los mismos son gestionados y suministrados por el servidor.
- No requieren instalación desde el cliente.
- Son multiplataformas. Sólo requieren un navegador compatible con la interfaz.
- Tienen un bajo consumo de recursos, ya que el volumen de datos más significativos se halla ubicado en un servidor.
- Tienen actualización inmediata.

Estas aplicaciones también presentan algunas desventajas que provienen de la naturaleza misma de la topología de la aplicación. Una desventaja crucial es la dependencia cruzada de los recursos utilizados. Es común que los archivos, imágenes, bases de datos y demás contenidos de la aplicación se hallen localizados en diversos equipos, incluso, ubicados en distintos espacios físicos. Esta distribución aumenta la probabilidad de que algún recurso se encuentre inhabilitado por algún motivo, pudiendo dejar inutilizado el servicio completo.

En la última década el acceso a internet se ha extendido exponencialmente y además se han diversificado enormemente los dispositivos que acceden a la misma. En la Figura 1.3 se presenta un análisis realizado por la empresa Microsoft, donde

[3]Cascade Style Sheet: Lenguaje utilizado para definir formato en archivos HTML y XML.
[4]Hypertext Transfer Protocol: Protocolo de transacción de la World Wide Web

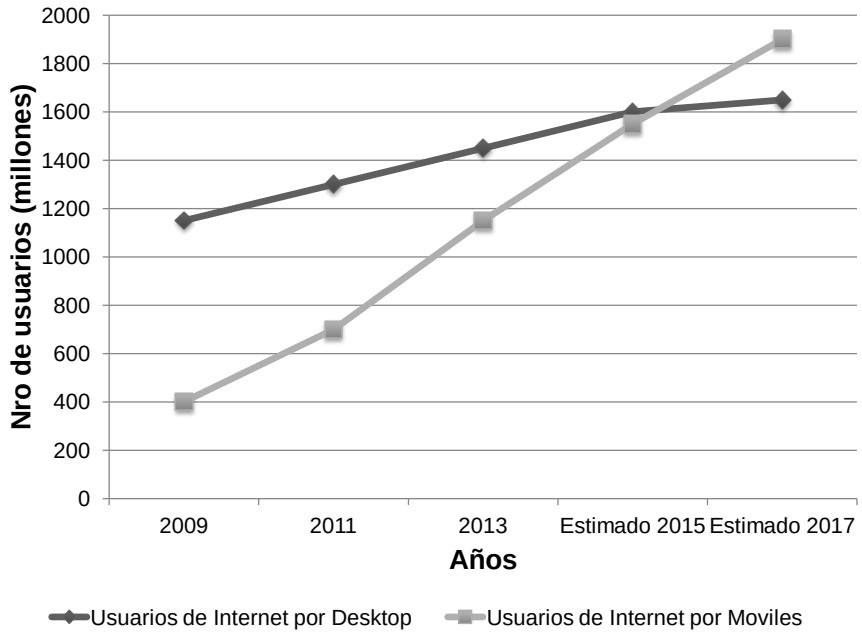


Figura 1.3: Evolución de distintos usos de dispositivos para el acceso a internet.[5]

puede verse la evolución de las ventas de equipos móviles y la proyección para 2015 de la cantidad de usuarios de acceden a internet a través de una computadora personal y a través de un dispositivo móvil. Actualmente la mayor cantidad de accesos se realiza desde dispositivos denominados clientes híbridos (dispositivos capaces de procesar datos pero con poca o nula capacidad de almacenamiento), como por ejemplo televisores inteligentes, celulares, tabletas y consolas portátiles. Esta tendencia potenció el desarrollo de las aplicaciones web, reemplazando en algunos casos, parcial o completamente a las aplicaciones de escritorio y servicios tradicionales.

Entre las aplicaciones web más relevantes se encuentran aquellas referidas a sistemas de información geográfica, servicios de multimedia, juegos y sistemas operativos nativos de internet [8, 9, 10, 11]. Todas estas aplicaciones están compuestas por varios servicios que integran un importante número de técnicas y procedimientos (entre ellas, el procesamiento digital de imágenes), aumentando la demanda de herramientas para el PDI en el campo del desarrollo web.

[5]Fuente: Microsoft Tag. http://software.intel.com/es-es/. Fecha Septiembre, 4, 2013

1.2.3. El servidor

Un servidor es un equipo que responde a los pedidos de los clientes. Este provee servicios de aplicaciones, de impresión y fax, de correo, de seguridad, administración de servicios de red, servicios de base de datos y otros relacionados con la manipulación de datos persistentes. Uno de los servidores más importantes que podemos encontrar en internet es el servidor web. Un servidor web es aquél que puede responder consultas de transacciones mediante protocolos soportados por la World Wide Web. Por lo general, el protocolo utilizado por los servidores web es el HTTP. Para poder manejar las operaciones básicas de este protocolo se pueden utilizar varias tecnologías que, mediante lenguajes de programación, ejecutan los comandos necesarios para establecer la comunicación con el cliente. Los lenguajes más utilizados son PHP, Ruby, Python y Java, aunque también pueden utilizarse comandos compilados de otras tecnologías, como C o C++ [2].

Cuando un cliente realiza una petición, el servidor debe consultar internamente si es capaz de responder a dicho pedido, en caso afirmativo pone en una cola la petición del usuario. Una vez que la respuesta es enviada al cliente, este debe procesarla y distribuirla en base al recurso que la solicitó.

Este tipo de tecnologías, ubicadas del lado servidor, poseen ciertas características en base al rol que constituyen en la aplicación web. De manera general se pueden mencionar las siguientes características de las herramientas del lado servidor:

- Operan con archivos persistentes.
- Están ocultas al usuario.
- Dependen de los recursos del procesador.
- No interactúan con el usuario.
- Deben ser instaladas en servidor.
- Permiten individualizar recursos para diferentes usuarios.

Las herramientas para el procesamiento de imágenes en el ámbito web más comúnmente utilizadas son herramientas ubicadas del lado servidor aunque, por las características mencionadas anteriormente, las mismas carezcan de interactividad, y no puedan ser utilizadas directamente por el usuario. Para el caso de algunas bibliotecas especializadas, las mismas deben ser instaladas en el servidor y contar con los permisos necesarios para su utilización.

1.2.4. El cliente

El cliente es un equipo que responde al usuario final, capaz de interactuar con el mismo interpretando los eventos que éste realice y efectuando peticiones al servidor.

En una aplicación web, el cliente se encarga de ejecutar la interfaz gráfica, interactuar con el usuario y transmitir sus consultas al servidor mediante el protocolo de hipertexto HTTP [4]. El cliente web es capaz de interpretar lenguaje de marcas (HTML), reglas de estilo (CSS), lenguajes de scripts, y aplicaciones cerradas.

Existen 3 tipos de clientes comúnmente definidos, los clientes livianos, híbridos y pesados. Los clientes livianos son aquellos que no pueden procesar datos ni tienen almacenamiento local, como algunos teléfonos y terminales bobas. Los clientes híbridos pueden procesar datos pero tienen poca o nula capacidad de almacenamiento, son ejemplos los smartphone, tabletas, televisores inteligentes y consolas. Los clientes pesados pueden almacenar y procesar datos de manera local, como una computadora con disco de almacenamiento.

Los dispositivos de clientes híbridos son, en la actualidad, los más utilizados para acceder a internet (Figura 1.3) aunque las redes móviles utilizadas en los mismos son de menor velocidad y consistencia que las redes usadas habitualmente en clientes pesados. Esta característica exige a los desarrolladores de aplicaciones evitar la comunicación constante y delegar procesamiento al cliente. Esta tendencia va desplazando a tecnologías como Java (en el lado cliente) y Flash, que requieren de complementos cerrados que no permiten su optimización o el manejo de recursos. Es por este motivo que los navegadores comienzan a optimizar la ejecución de tecnologías estándares del lado cliente, principalmente el compilador del lenguaje interpretado JavaScript (JIT[6]). En la Figura 1.4 se puede observar la evolución del rendimiento de los JIT de JavaScript a través de las diferentes versiones en los tres principales navegadores del mercado. Las evaluaciones fueron realizadas con los test Benchmark para Google Chrome (Figura 1.4(a)), Sunspider para Internet Explorer (Figura 1.4(b)) y Octane para Firefox (Figura 1.4(c)). Los test miden el tiempo utilizado por el equipo para ejecutar rutinas particulares, como algoritmos de compresión, algoritmos de encriptación, etc. Cada test realiza un promedio ponderado con las diversas rutinas y obtiene un coeficiente que es el utilizado para comparar diferentes equipos. Todos los resultados aquí citados fueron extraídos del sitio: https://www.cpubenchmark.net/.

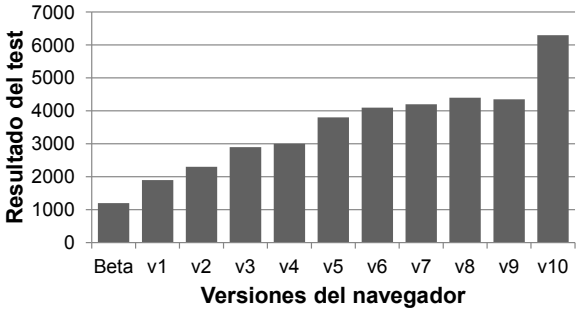
En el 2009, la World Wide web Consortium (W3C[7]), estableció una serie de nuevos elementos para la quinta revisión de HTML (HTML5). CANVAS es un frame de renderizado que implementa primitivas de dibujo, una versión adaptada del motor gráfico OpenGL[8] y permite visualizar y procesar imágenes y videos. Este frame de renderizado es dinámico, es decir, no hace falta recargar la página para actualizar o modificar su contenido. La etiqueta CANVAS permite operar con registros de 64 bits, haciendo posible que los JITs de los navegadores puedan realizar procesamiento paralelo a nivel hardware, reduciendo de manera significativa el tiempo de procesamiento y la carga de memoria [19, 20]. Este elemento brinda una gran varie-

[6]Just In Time compilation: Compilador de JavaScript ubicado en los navegadores

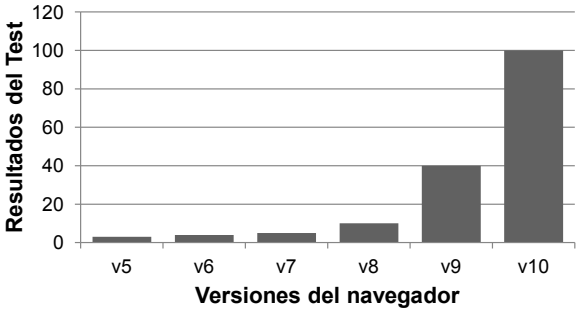
[7] *World Wide Web Consortium.*. Consorcio internacional que produce recomendaciones para la World Wide Web.

[8]https://www.opengl.org/

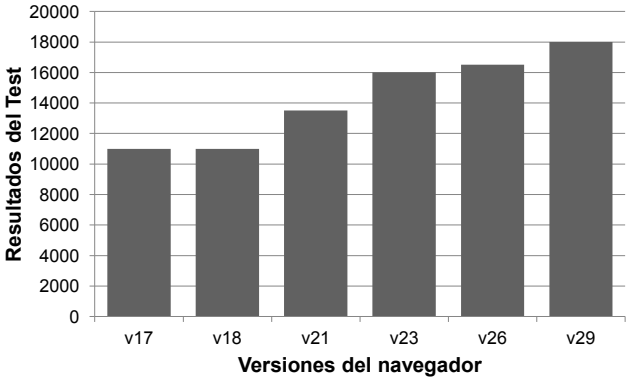
dad procesamientos de media y alta complejidad en el navegador, que no afecta el comportamiento del software, beneficiado a los dispositivos de menor rendimiento. Además, al ser un estándar de una tecnología multiplataforma, es posible utilizar el mismo desarrollo para diversos dispositivos sin necesidad de instalar complementos adicionales. Cabe destacar que esta tecnología aun no ha sido explotada para desarrollos de PDI, ya que su aparición es reciente y la misma se encuentra en etapas de aprobación.



(a) Test para Chrome.



(b) Test Internet Explorer.



(c) Test para Firefox.

Figura 1.4: Evolución de rendimiento de JIT para distintos navegadores.

1.3. Objetivos del Proyecto Final

Definida la motivación y el entorno general en el que se enmarca el trabajo, se exponen a continuación los objetivos generales y específicos del presente Proyecto Final, además se incluyen los alcances del mismo.

1.3.1. Objetivos generales

Crear un conjunto de utilidades para el procesamiento de imágenes capaz de desempeñarse en cualquier dispositivo que posea un navegador compatible con el estándar HTML5, conformando una biblioteca de métodos, desarrollada con tecnologías web estándar del lado cliente del modelo Cliente - Servidor.

1.3.2. Objetivos específicos

- Elaborar un relevamiento sobre los actuales recursos tecnológicos de las tecnologías web clientes, enfocando la búsqueda hacia sistemas que utilicen procesamiento de imágenes.
- Realizar un patrón de diseño que complemente las primitivas nativas de la tecnología con las funciones de procesamiento desarrolladas para el proyecto y que conserve el buen rendimiento de la tecnología.
- Proponer una interfaz que facilite la integración de la biblioteca en otros proyectos y que acelere la curva de aprendizaje de manejo la misma.
- Desarrollar métodos de procesamiento de imágenes y extender su aplicación a capturas de video.
- Evaluar el desempeño de la biblioteca: precisión, potencialidad de uso en múltiples navegadores.

1.3.3. Alcances

En este proyecto se desarrolla una biblioteca para el procesamiento de imágenes con tecnologías web estándar del lado cliente, de código abierto, que permita ser utilizada en diversos dispositivos sin comprometer el rendimiento del mismo. Con una estructura clara y bien constituida con el fin de ser cedida a la comunidad para su enriquecimiento.

Dentro de los alcances del proyecto se pueden enumerar los siguientes:

- El sistema se enfoca en el procesamiento de imágenes, sin embargo, se definirán estructuras genéricas que permitan el manejo de otro tipo de archivos. Pruebas preliminares sobre video en tiempo real son incluidas de forma complementaria.

- Los métodos de transformación frecuencial se limitarán a imágenes con dimensión de potencia de 2.
- Un diseño escalable para su continuidad en trabajos futuros.
- No se pretende realizar una aplicación final específica y completa orientada al usuario final para demostrar el uso del proyecto.
- No se realizarán exhaustivas comparaciones de rendimientos con otras bibliotecas de tecnologías de naturaleza distinta.

CAPITULO 2

Fundamentos teóricos

En este capítulo, se presentan los fundamentos teóricos sobre los cuales se sustenta el proyecto. Estos fundamentos se pueden encontrar en diferentes libros de PDI. En este trabajo se siguen las definiciones de "R. Gonzalez and R. Wood, Digital Image Processing, 2do Edition" [17]. Primeramente se describen las operaciones de realce sobre las imágenes, seguido de las operaciones de filtrado en dominio espacial, frecuencial, descripciones del modelo de color, finalmente operaciones morfológicas y segmentación sobre las imágenes. El capítulo concluye con las características teóricas de la tecnología con la cual fue desarrollado el proyecto.

2.1. La imagen digital

Una imagen digital o analógica digitalizada se representa mediante una función de dos variables *f(x,y)*, donde x e y son las coordenadas de un elemento de la matriz de N por M componentes, siendo N y M el ancho y alto de la imagen, respectivamente. El valor de la función en una coordenada particular corresponde a un número real que representa el nivel de gris (para una imagen en escala de grises), una terna de valores reales para una imagen en modelo de color tricanal (RGB, HSI, etc), o múltiples valores, en caso de ser una imagen satelital. De esta manera, la función de la imagen queda definida por:

$$f(x,y) = \begin{pmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{pmatrix}. \qquad (2.1)$$

2.2. Realce de imágenes en el dominio espacial

Los métodos de realce en el dominio espacial son procedimientos orientados a la tarea de mejoras de detalles o iluminación en las imágenes. Estos métodos operan directamente sobre los píxeles de la imagen y pueden denotase con la expresión:

$$g(x, y) = T[f(x, y))], \tag{2.2}$$

donde *f(x, y)* es la imagen de entrada, *g(x, y)* la imagen resultado de la operación y T un operador sobre f definido sobre alguna vecindad alrededor de *(x, y)*.

El concepto de vecindad en un punto *(x, y)* refiere a una zona de la imagen (cuadrada o rectangular) centrada en *(x, y)*. Si la vecindad es de 1x1 (un píxel), *g(x,y)* depende únicamente del valor de f en *(x, y)*.

En (2.2) la función T puede denotar distintos tipos de funciones, como por ejemplo, funciones matemáticas, funciones aplicadas a todo el rango de valores, funciones definidas por partes, etc.

2.3. Filtrado de imágenes en el dominio espacial

El filtrado espacial es una operación que permite enfatizar o atenuar bordes. Opera modificando el valor de cada pixel en base a una vecindad, por lo que se la considera una operación de transformación local.

2.3.1. Filtrado lineal

El filtrado lineal de una imagen f de N por M se realiza mediante la convolución de ésta con una imagen "máscara" w de n por m y está definida por:

$$g(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x - s, y - t), \tag{2.3}$$

donde $a = \frac{(m-1)}{2}; b = \frac{(n-1)}{2}; x = [0, M - 1]; y = [0, N - 1]$.

La ecuación (2.3) se conoce como convolución. Las diferentes máscaras de filtrado permiten obtener diversos resultados sobre la imagen.

Filtrado pasa altos

Los filtros pasa altos permiten realzar los detalles de una imagen como bordes o discontinuidades mediante el enfatizado de las altas frecuencias y la atenuación

de las áreas de baja varianza de los niveles de grises. Tanto el factor de enfatizado como el de atenuación pueden ajustarse empleando distintas máscaras.

Si la suma de los coeficientes de la máscara w es 1, se realzan las altas frecuencias conservándose el brillo medio de la imagen sobre la cual se aplica el filtro. Si la suma de los coeficientes de la máscara w es cero, se realzan las altas frecuencias pero se elimina el brillo medio. Los coeficientes de la máscara poseen pesos positivos y negativos.

Filtrado pasa bajos

Los filtros pasa bajos permiten reducir picos y transiciones rápidas de niveles de grises. El efecto de los filtros pasa bajo es el de un promediado de los píxeles afectados por la máscara de convolución.

Todos los coeficientes de la máscara w deben ser positivos y la suma de los mismos igual a 1 para mantener el brillo medio.

2.3.2. Filtrado no lineal

Los filtros estadísticos de orden son filtros espaciales no lineales, cuya respuesta se basa en el ordenamiento de los valores de los píxeles. El valor de cada pixel en la imagen destino estará dado por el resultado de aplicar alguna operación a la vecindad del mismo. Algunas de estas operaciones son: mediana, máximo, mínimo, moda.

2.4. Filtrado en el dominio frecuencial

El filtrado en el dominio frecuencial permite modificar las características del espectro de frecuencia de una imagen. El motivo por el cual se utiliza un filtro en frecuencia es para poder operar con características de la imagen que en su dominio original resultan difícil de extraer o discriminar. Para expresar una imagen en dominio frecuencial se le aplica a la misma una transformación de Fourier.

Sea *f(x,y)* una imagen de dimensión N por M . Se define la transformada de Fourier de la siguiente manera:

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(ux/M+vy/N))}, \tag{2.4}$$

siendo u y v el conjunto de coordenadas en el espacio de Fourier. De la misma manera se expresa su inversa:

$$f(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) e^{j2\pi(ux/M+vy/N))}. \tag{2.5}$$

El procedimiento para realizar un filtrado en el dominio de Fourier es:

- Aplicar la transformación directa de Fourier a la imagen original.
- Aplicar una máscara de filtrado mediante multiplicación punto a punto.
- Aplicar transformación inversa de Fourier.

2.5. Procesamiento de color

El uso del color en el procesamiento de imágenes es motivado por dos factores fundamentales. Primero, el color es un elemento de descripción que permite identificar y extraer objetos de una escena. Segundo, el ojo humano puede discernir mayor cantidad de variaciones de colores, en comparación con las que puede distinguir en niveles de grises.

Un color se representa como un punto en un sistema de coordenadas, cuyas características están definidas por un modelo. Los distintos modelos están orientados a distintos dispositivos de hardware o a facilitar los procesamientos basados en la percepción humana.

2.5.1. Modelo RGB

El modelo RGB permite representar cualquier color en función de los colores rojo, verde y azul. Este modelo se basa en un sistema cartesiano, y cada dimensión está representada por los valores reales entre 0 y 1. La precisión estará dada en base al tipo de datos escogido para representar los elementos de la imagen. El modelo RGB es un modelo aditivo, es decir, los colores se sintetizan mediante la suma de sus elementos. Este modelo es usado por la mayoría de los formatos digitales.

En la Figura 2.1 se puede ver el modelo RGB representado por el cubo que abarca el octante conformado por los ejes Rojo, Verde y Azul. La diagonal que va desde el *(0,0,0)* al *(1,1,1)* representa la escala de grises.

2.5.2. Modelo HSI

El modelo HSI permite representar los colores mediante componentes de Tono, Saturación e Intensidad que se ajustan mejor al sistema visual humano. El modelo

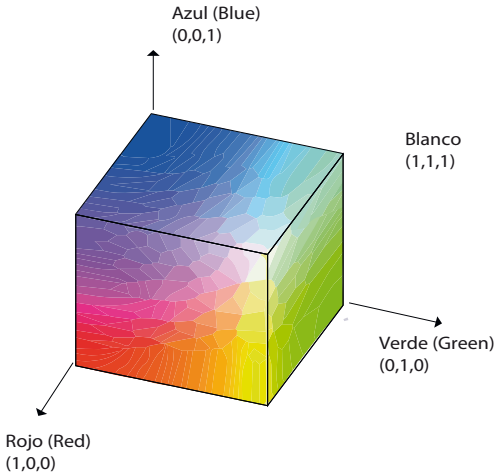


Figura 2.1: Modelo RGB

HSI no tiene una única representación gráfica asociada, aunque se la suele asociar a la de un cono circular doble, como se aprecia en la Figura 2.2. Dado que la componente de intensidad está desacoplada de la información de color de la imagen, este modelo suele emplearse fundamentalmente en algoritmos basados en las propiedades de percepción humana.

2.6. Operaciones morfológicas

El procesamiento morfológico permite destacar la estructura de los objetos que constituyen la imagen. Se suele utilizar como pre-procesamiento para simplificar formas, extraer estructuras, describir objetos (áreas, perímetros, etc).

Un conjunto de operaciones muy utilizadas son la erosión y la dilatación. Su implementación se realiza a partir de la comparación de una vecindad alrededor de un pixel de la imagen original y una máscara (elemento estructurante). Ambas operaciones son utilizadas comúnmente con imágenes binarias como un pre-proceso, aunque también se definen para imágenes en escalas de grises.

Dilatación

Sea f una imagen, w un elemento estructurante y g una vecindad alrededor del pixel *f(x,y)* en un punto *(x,y)*. El valor resultante del pixel *f(x,y)* corresponde al mayor valor de los elementos de la vecindad g que estén incluidos completamente en el solapamiento entre la vecindad g y el elemento estructurante w.

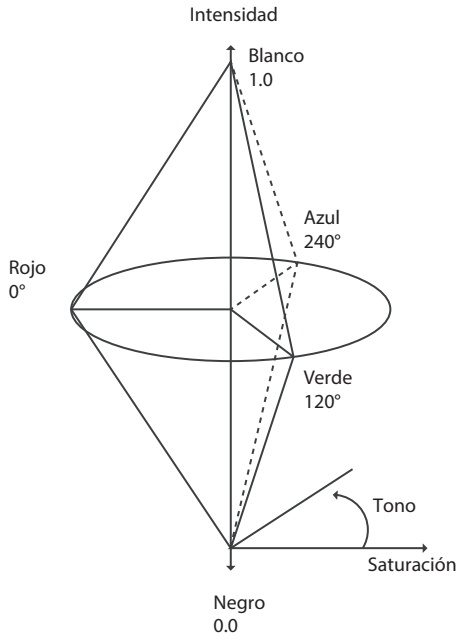


Figura 2.2: Modelo HSI

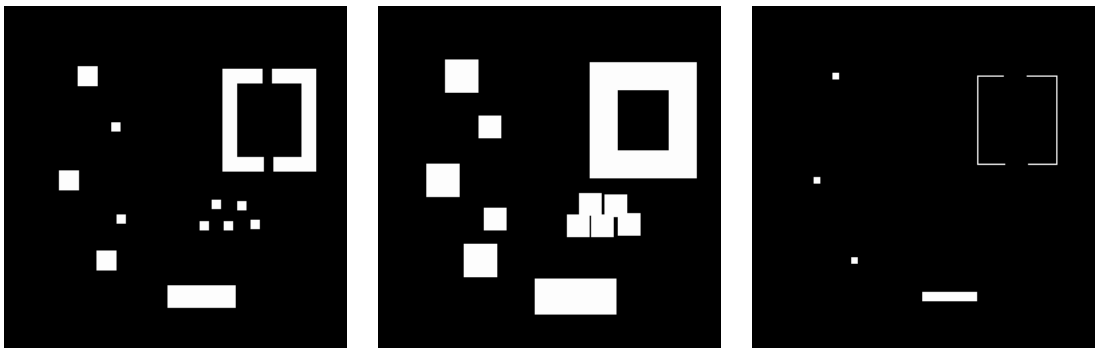
En la Figura 2.3(b) se ve el resultado de la dilatación de la Figura 2.3(a) con el elemento estructurante de la ecuación (2.6). La dilatación permite unir regiones o bordes desconectados.

Erosión

Sea f una imagen, w un elemento estructurante y g una vecindad alrededor del pixel *f(x,y)* en un punto *(x,y)*. El valor resultante del pixel *f(x,y)* corresponde al menor valor de los elementos de la vecindad g que estén incluidos completamente en el solapamiento entre la vecindad g y el elemento estructurante w.

En la Figura 2.3(c) se ve el resultado de la erosión de la Figura 2.3(a) con el elemento estructurante de la ecuación(2.6). La erosión permite eliminar detalles irrelevantes o ruido aislado.

$$w = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \tag{2.6}$$



(a) Imagen binaria con elementos de diversas dimensiones y formas

(b) Dilatación

(c) Erosión

Figura 2.3: Dilatación y Erosión de imagen binaria

2.7. Segmentación de imágenes

La técnicas de segmentación permiten obtener los componentes de una imagen: objetos, subregiones, etc. Los algoritmos de segmentación se basan en agrupar los píxeles contiguos que comparten cierta característica, por lo general, mismos niveles de grises o bien saltos de discontinuidad que signifiquen posibles bordes.

Para la detección de discontinuidades se utiliza una máscara que se desplaza por la imagen mediante el procedimiento descrito en la sección 2.3. La respuesta R a la máscara en cualquier punto de la imagen está dada por:

$$R = \sum_{i=1}^{n} w_i z_i. \tag{2.7}$$

, siendo n la cantidad de elementos de la máscara, w_i son los coeficientes de la máscara y z_i son los niveles de grises de la imagen asociados al coeficiente de la máscara para esa varianza.

Detección de puntos y líneas

La detección de un punto aislado se basa en la suposición de que existe un quiebre abrupto en las tonalidades de grises entre un pixel y su vecindad. Es posible detectarlo con una máscara de respuesta 0 (ver ecuación 2.8). Cuando la sumatoria de los elementos de la máscara es 0, la respuesta a áreas de grises constantes también lo es.

$$w = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}. \tag{2.8}$$

Con el mismo concepto que los puntos, se plantean máscaras capaces de detectar líneas en base a la respuesta obtenida a partir de la ecuación 2.7. En la ecuación 2.9 se ve una máscara para detección de lineas horizontales. Rotando dicha máscara es posible obtener las líneas que existen en diferentes direcciones.

$$w = \begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix}. \tag{2.9}$$

Detección de bordes mediante primera derivada

El método de la primera derivada permite detectar el cambio de tonalidades del borde de una imagen mediante la aplicación de la operación gradiente de las variables x, y definido como:

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \\ \frac{\partial f}{\partial y} \end{bmatrix}. \tag{2.10}$$

Existen varios operadores que implementan la ecuación (2.10), cada uno tiene sus ventajas y desventajas dependiendo del caso de uso. Los más comúnmente usados son: Roberts, Prewitt y Sobel

Detección de bordes mediante segunda derivada

El método de la segunda derivada permite detectar bordes a partir de la identificación del signo de la pendiente a un lado y el otro del borde. El operador Laplaciano, definido en la ecuación 2.11, permite aproximar la segunda derivada.

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}. \tag{2.11}$$

2.8. Características de la tecnología utilizada

A continuación se describen algunas características básicas de la tecnología utilizada, comenzando por los lenguajes HTML y JavaScript. Posteriormente, se describen algunos puntos particulares sobre estas y que son de especial consideración para el desarrollo de este proyecto.

HTML

HTML es un protocolo de hipertexto utilizado para desarrollar páginas web, admite una serie de lenguajes que permiten crear interacción entre el usuario y en contenido del documento de forma instantánea y dinámica. En la revisión 5 del protocolo, se incorpora el elemento CANVAS, un lienzo capaz de manipular gráficos vectoriales, renderizar videos, imágenes y que posee internamente una versión reducida del motor gráfico OpenGL (OpenGL ES) [18, 19].

JavaScript

JavaScript es un lenguaje de scripting[1] normado por la ECMAScript[2], basa su sintaxis en el lenguaje C++. JavaScript es un lenguaje no tipado y orientado a eventos, no manipula punteros, sin embargo, opera todas sus variables por referencias. Es el lenguaje más utilizado para manipular el script de HTML [6].

Optimizaciones sobre JavaScript

La quinta revisión de HTML incorpora, además de etiquetas capaces de manipular elementos multimedia, optimizaciones sobre ciertas estructuras que permiten a los motores de los navegadores transferir gran parte del procesamiento al hardware. Para aprovechar esa optimización se incorporaron, en JavaScript versión 1.5, dos características que posibilitan a los JITs minimizar el uso de la memoria: la posibilidad de definir un tipo de dato específico a las variables para todo el ámbito y la incorporación de un tipo particular de arreglos denominados "arreglos explícitamente tipados". Estos son una serie de estructuras que alojan, exclusivamente, datos de cierta precisión en bits y poseen un grupo de métodos que permiten manipular subconjuntos de datos en bloques mediante referencias, sin necesidad de duplicar información [18].

Características técnicas del objeto CANVAS

Para la manipulación de imágenes, el elemento HTML - CANVAS, posee una estructura interna que permite el acceso a los píxeles del lienzo. Esta estructura es un arreglo unidimensional de 4 canales por pixel, de 8 bits de precisión por cada canal (32 bits por pixel). Cada uno de estos canales trabaja en un rango entre 0 y 255 e implementa el modelo de color RGBA[3]. El objeto utilizado para contener esta información es un arreglo explícitamente tipado de tipo uint8ClampedArray. Cada elemento CANVAS tiene asociado, recíprocamente, un único arreglo [18].

[1] Lenguaje interpretado cuya compilación se realiza *al vuelo*

[2] http://www.ecma-international.org/publications/standards/Ecma-262.htm

[3] Modelo de color derivado del modelo RGB al cual se le agrega un canal *alpha* que permite controlar la opacidad o transparencia del color RGB final.

CAPITULO 3

Desarrollo de la biblioteca propuesta

En este capítulo se presenta detalladamente el diseño realizado para la biblioteca, se describe el desarrollo en base al paradigma de programación orientado a datos, el patrón de diseño JavaScript y la relación entre ambos.

3.1. Diseño general de la biblioteca

Para el diseño de la estructura de la biblioteca se utilizó un diagrama de Paquetes UML [1]. Dado que un proyecto de estas características carece de una interfaz de interacción y de clases interrelacionadas, no es posible utilizar un diseño estándar de ingeniería de requerimientos. No obstante, el diseño posee un flujo de información complejo entre los métodos principales y una estructura de datos que necesita ser reflejada de forma gráfica [12, 3].

Inicialmente, se pensó en realizar el desarrollo de la biblioteca de forma íntegra, diseñando incluso su estructura. Sin embargo, luego de un análisis profundo de la librería OpenCV para C se decidió utilizarla como referencia en base a dos motivos principales. El primer motivo es que OpenCV es una de las herramientas de procesamiento de imágenes más utilizada y por lo tanto, su estructura ha sido testeada por una vasta comunidad de desarrolladores. En segundo lugar, se consideró el hecho de que existen muchos proyectos realizados con ésta, así como también un gran número de usuarios desarrolladores. Entonces, manteniendo una estructura e interfaz similar, es factible despertar el interés de los desarrolladores de OpenCV, mientras que las migraciones de proyectos consumados es posible realizarla con mínimos ajustes. OpenCV es una biblioteca desarrollada por la empresa Intel que permite trabajar en lenguajes como C, C++ y Python y exportar los resultados a aplicaciones de escritorio y móviles con sistema operativo Android [13, 14]. Si bien el diseño y la

[1] Unified Modeling Language, en español Lenguaje Unificado de Modelado

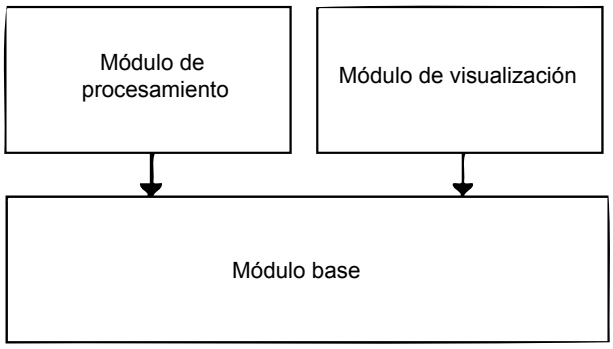


Figura 3.1: Diagrama de interacción entre módulos

interfaz de la biblioteca están basados en OpenCV para C, el flujo de información está desarrollado en base al diseño propuesto para la biblioteca, y el comportamiento interno de los métodos de procesamiento fue implementado, en su mayoría, mediante definición teórica.

La estructura de la biblioteca desarrollada se organiza como un diagrama modular, donde cada elemento integrante se encuentra agrupado en un módulo según su funcionalidad. Los módulos no tienen dependencia mutua, se relacionan únicamente en la etapa de aplicación. Para la implementación se utilizó paradigma de programación Programación Orientada a Datos. Este se desprende del paradigma orientado a objetos (POO), pero su estructura se centra en la comunicación directa entre los datos.

La estructura de la biblioteca está organizada en 3 módulos principales. El **módulo de procesamiento** está integrado por funciones en las que se implementan algoritmos de procesamiento específicos de PDI, como ser: convolución de imágenes, filtrado de imágenes, transformada de Fourier, transformaciones de modelo de color, etc. Además se incluyen funcionalidades más generales, como ser: cambio de nivel de profundidad de bits del pixel, selección de cantidad de canales de la imagen, entre otras. En el **módulo base** se encuentra definida la estructura principal, las estructuras secundarias y las constantes. La estructura principal está compuesta por una colección de datos ordenados, entre los mismos se encuentra un arreglo con los valores de los píxeles de la imagen y un conjunto de metadatos que hacen a la definición de la imagen: el alto, el ancho, la profundidad de bits del pixel, el modelo de color utilizado, la cantidad de elementos, la condición de borde utilizada e información adicional para la aplicación de los métodos. Las estructuras secundarias están integradas por elementos particulares, definidos exclusivamente para el uso interno de la estructura principal, como por ejemplo el objeto AIROI (Que permite definir una región de interés). Al **módulo de visualización** lo integran los métodos de salida o interface. Estos métodos permiten interpretar gráficamente los datos de la estructura principal para exportarlos en formatos de imágenes.

En la Figura 3.1 se puede ver el diagrama de interacción entre módulos. Los

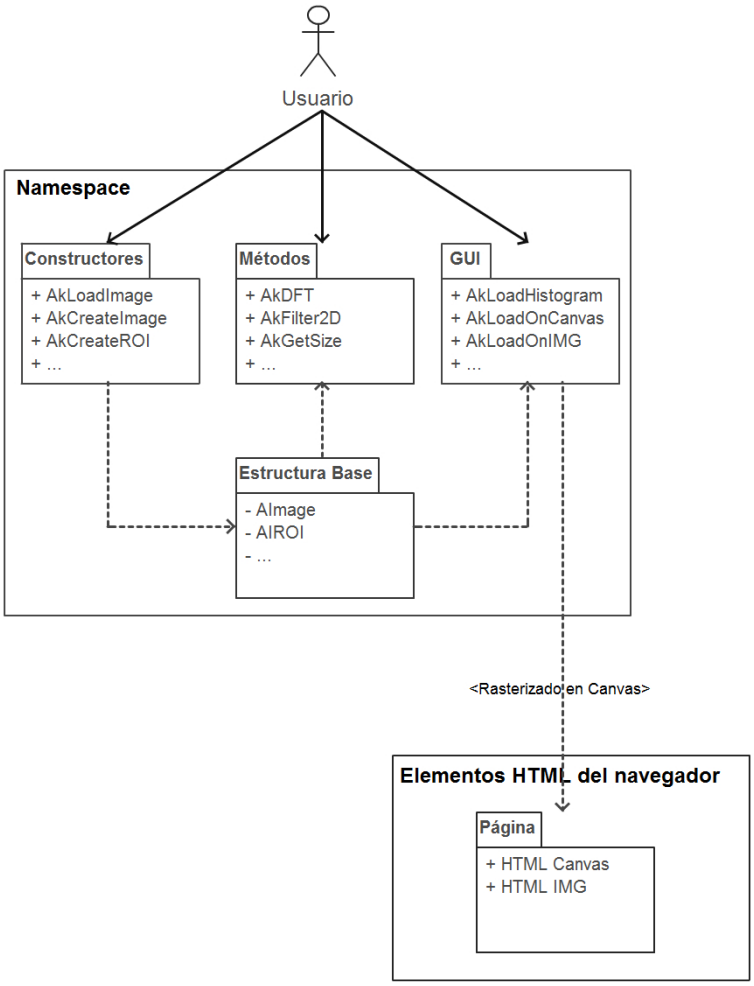


Figura 3.2: Diagrama UML del diseño general de la biblioteca

módulos de procesamiento y de visualización interactúan exclusivamente con el módulo base. Las estructuras generadas por el módulo base se utilizan como entrada en los módulos restantes. Los datos obtenidos a partir en el módulo de procesamiento pueden ser utilizados nuevamente por éste o bien exportados por el módulo de visualización.

El diagrama de paquetes UML permite mostrar casos de relaciones generales y adaptar a cualquier tipo de estructura que requiera reflejar una interacción entre sus partes. En la Figura 3.2 se observa el diagrama de paquetes UML correspondiente al diseño de la biblioteca. En la parte superior se encuentra el actor (usuario-programador) que tiene acceso a todos los elementos y atributos de la biblioteca. Todos los métodos se encuentran distribuidos de forma independiente en un mismo namespace[2]. Con los métodos constructores es posible crear a partir de un conjunto de parámetros las estructuras: AImage (estructura base), AIROI (Región

[2]Contenedor abstracto que permite unificar elementos bajo un mismo domino.

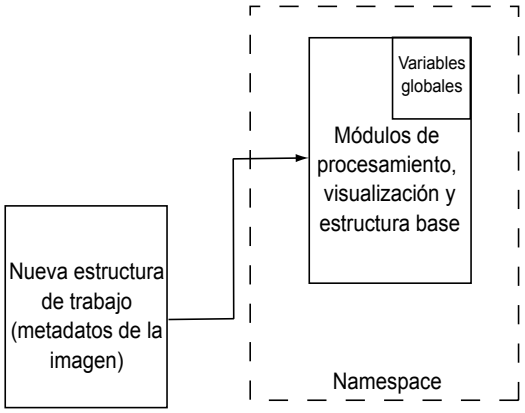
de interés) y AKHistogram (Histograma). Estas estructuras son utilizadas como elementos de entrada por los métodos de procesamiento. Los métodos de visualización permiten exportar los datos a distintos formatos de imágenes o rasterizarlos sobre los elementos HTML: IMG o CANVAS.

3.2. Patrón de diseño empleado

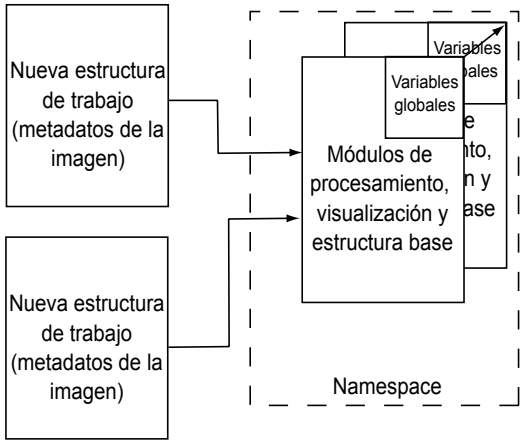
Los patrones plantean soluciones que se han demostrado efectivas a problemas comunes. Existen dos tipos bien diferenciados de patrones, los patrones de diseño y los patrones de programación. Los primeros plantean esquemas y conceptos que sirven para resolver problemas comunes de diseños de software. Los patrones de programación ofrecen soluciones efectivas a situaciones comunes, en el desarrollo de software, para algún lenguaje de programación particular [15].

La interfaz del proyecto se diseñó en base a la interfaz de la biblioteca OpenCV para el lenguaje C, utilizando un paradigma de Programación Orientada a Datos mediante un diseño modular que ubica a los elementos dentro de un namespace que protege los métodos de la duplicación de nombres. El conjunto de métodos es independiente del procesamiento utilizado y no se modifican a lo largo de su utilización, por lo que el namespace que los contiene debe ser instanciado una única vez. Para cumplir con esta característica se optó por utilizar el patrón de desarrollo Singleton que permite restringir la creación de objetos de una clase o valores de un tipo a un objeto único [16, 15]. El objetivo de este patrón consiste en garantizar que una clase o variable tenga solo una instancia y proporcionar un punto de acceso global a ella. En cada instanciación de una nueva estructura el conjunto de métodos se conserva, sin embargo, esto no permite operar con más de una estructura (imagen) de forma simultánea. Para permitir múltiples instancias de la estructura base y mantener la característica principal del patrón Singleton, se agregaron instrucciones del patrón de diseño “Prototipo” que permite la instanciación de objetos a partir de la copia de un objeto original. El patrón resultante permite operar con múltiples imágenes y mantiene una única instancia del conjunto de métodos.

En las Figuras 3.3(a) y 3.3(b) se aprecia el esquema de funcionamiento de la biblioteca con el patrón Singleton. En la Figura 3.3(a) se puede ver la creación de una instancia de la estructura principal a partir de los constructores, cargando en memoria el namespace junto a todos los métodos de la biblioteca y las variables globales. En la Figura 3.3(b) se crea una segunda estructura, esta instanciación sobreescribe los módulos de contenido estático y conserva las variables globales. Finalmente, la nueva estructura se crea a partir de la copia de la estructura base original.



(a) Primer instanciación del namespace



(b) Segunda instanciación del namespace

Figura 3.3: Comportamiento de la instanciación del namespace

3.3. Diagrama de flujo del algoritmo general de procesamiento

Este proyecto está realizado íntegramente con herramientas web del lado cliente. Para su utilización es necesario contar con un documento HTML, al menos un elemento CANVAS-HTML y una etiqueta de script en el mismo archivo o en un archivo separado.

La Figura 3.4 ilustra el diagrama de flujo del algoritmo general para un procesamiento completo realizado con la biblioteca. El diagrama permite definir una Región de Interés, generar un Histograma, realizar un procesamiento y visualizar los resultados. El primer paso consiste en la creación de una estructura AImage, que contiene toda la información de la imagen (o lienzo). La estructura puede crearse

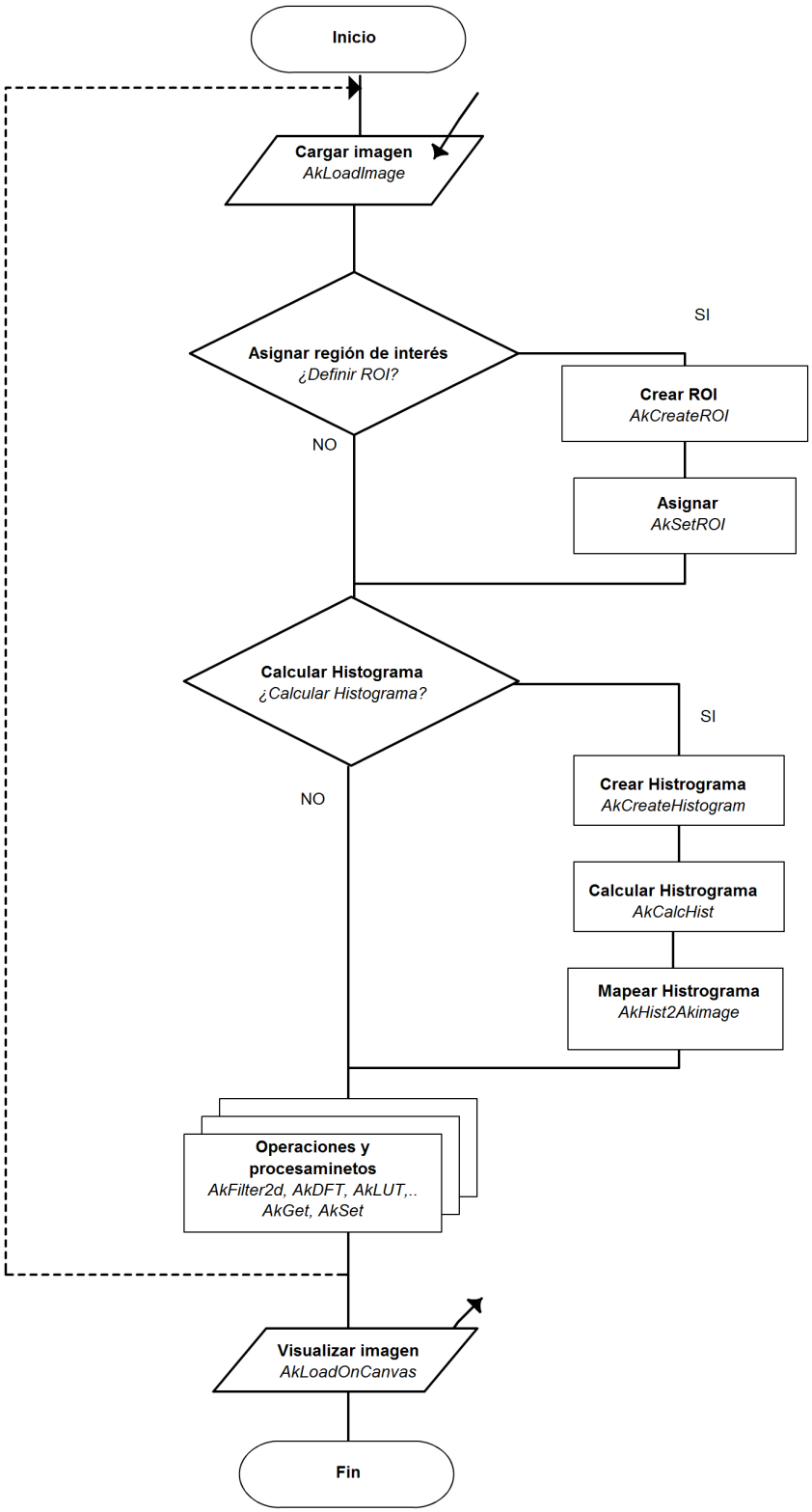


Figura 3.4: Flujo del algoritmo general de procesamiento

a partir de un lienzo vacío, de un archivo imagen o un objeto CANVAS con valores asociados. Para crear una estructura vacía se utiliza el método AkCreateImage, especificando el tamaño, la profundidad y la cantidad de canales (en base a las características de la imagen deseada). También es posible cargar una imagen y mapear sus valores a la estructura AImage. Para ello se utiliza el método AkLoadImage, especificando la referencia a la imagen y modelo de color a emplear. La referencia a la imagen puede ser una URL a un archivo con un formato compatible para el navegador[3], un objeto HTML del tipo imagen (HTML-IMG), un objeto HTML del tipo CANVAS, o sus respectivos ID, para ambos elementos. En cualquiera de los casos, el usuario, podrá referenciar la imagen utilizando la declaración efectuada en el documento HTML o crearla dinámicamente desde el script.

Para cada AImage, es posible definir una región de interés donde aplicar las distintas operaciones. Mediante el método AkCreateROI se crea la región, especificando el ancho, alto y desplazamiento, con el método AkSetImageROI se asocia a la estructura AImage. Posterior a la definición de la ROI es posible, también, calcular el histograma de una estructura AImage. Con el método AkCreateHist se crea la estructura histograma AkHistogram, especificando la cantidad de intervalos, con AkCalcHist se calcula para una estructura AImage dada, AkHist2Akimage permite mapear los valores del histograma a una estructura AImage. Tanto la definición de una ROI como la generación de un histograma son pasos de configuración opcionales y previos a la realización de cualquier operación de procesamiento. Instanciada la estructura AImage, puede aplicarse alguna operación predefinida como por ejemplo AkFilter2d, AkDFT, AkLUT o alguna operación puntual accediendo a los píxeles de la imagen mediante los métodos AkSet y AkGet.

Para visualizar los resultados se debe utilizar el método AkLoadOnCanvas. Este método mapea todos los datos de la estructura AImage a un elemento HTML-CANVAS, respetando la dimensión, canales y modelo de color. Las operaciones realizadas sobre el elemento CANVAS no requieren recargar la página, por lo tanto, si se utiliza el mismo elemento para visualizar dos estructuras diferentes, se mostrará aquella modificada en última instancia.

El procedimiento descrito para la Figura 3.4 detalla el algoritmo general donde se incluyen todas las operaciones. El diagrama marca la línea de procedimiento estándar y la línea de puntos indica que se puede retomar el proceso desde la carga de la imagen y repetirlo todas las veces que el usuario lo requiera. De esta manera, es posible generar múltiples imágenes procesadas y realizar operaciones entre ellas. Si se desea visualizar varias imágenes, cada estructura debe mapearse sobre un elemento CANVAS particular.

[3]Por defecto los navegadores soportan los formatos estándares (JPEG, PNG, GIF, PDF, SVG). Pero es posible utilizar codecs que le permitan al navegador interpretar otros formatos

3.4. Métodos implementados

A continuación se presentan los métodos desarrollados para la biblioteca.

Método: AkCreateImage

Descripción: AkCreateImage permite crear una estructura AImage vacía.
Parámetros:
size: Arreglo de dos elementos con el ancho y alto de la imagen.
depth: Precisión de los datos de la estructura. Puede tomar los siguientes valores:

- DEPTH_ 8U: 1 byte sin signo (0, 255). Profundidad por defecto del HTML CANVAS
- DEPTH_ 8S: 1 byte con signo (-127, 128)
- DEPTH_ 16S: 4 bytes con signo
- DEPTH_ 32S: 4 bytes sin signo
- DEPTH_ 32F: 4 bytes flotantes
- DEPTH_ 64F: 8 bytes flotantes

channels: Cantidad de canales de la imagen. Puede tomar los siguientes valores.

- 1 : 1 Canal (Escala de grises)
- 3 : 3 Canales (Modelos RGB)
- 4 : 4 Canales (Modelos RGBA)

Retorno: Estructura AImage

Método: AkLoadImage

Descripción: AkLoadImage permite crear una estructura AImage extrayendo los datos de una imagen u objeto.
Parámetros:
ImageReference: Referencia de donde extraer los datos para crear la estructura. Los tipos de datos que puede tomar son:

- Referencia a un objeto HTML CANVAS
- Referencia a un objeto HTML IMG
- URL de una imagen
- ID de un objeto HTML CANVAS

- ID de un objeto HTML IMG

isColor: Define el modelo de color con el cual será creada la estructura

- LOAD_ IMAGE_ COLOR : Modelo RGB
- LOAD_ IMAGE_ GRAYSCALE : Escala de grises
- LOAD_ IMAGE_ ANYCOLOR : Modelo RGBA

Retorno: Estructura AImage

Método: AkCreateROI

Descripción: AkCreateROI permite crear un objeto de AROI. Origen de la imagen en el eje izquierdo superior en la coordenada 0,0
Parámetros:
xOffset: Coordenada origen de la ROI en el eje x
yOffset: Coordenada origen de la ROI en el eje y
Width: Ancho de la ROI
Height: Alto de la ROI
Retorno: Estructura AROI

Método: AkSetImageROI

Descripción: AkSetImageROI asigna una estructura AROI en una estructura AImage.
Parámetros:
ImIn: Estructura AImage a la cual se desea asignar la AROI (pasaje por referencia)
ROI: Estructura AROI para ser asignada
Retorno: Estructura AImage (opcional, devuelve al referencia)

Método: AkCreateHist

Descripción: AkCreateHist permite definir una estructura AkHistogram.
Parámetros:
bins: Donde bins es el rango del histograma. Los bins deben definirse con vectores que especifiquen los rangos del intervalo de acción, es decir, el valor mínimo y máximo que tomará el rango del intervalo. Puede definirse un rango único o bien un rango de varios intervalos
Retorno: Estructura AkHistogram

Método: AkCalcHist

Descripción: AkCalcHist calcula los valores del histograma en base a lo definido en AkCreateHist y a la estructura AImage pasada por parámetro. Soporta AROI
Parámetros:
InAkimage: Estructura AImage con profundidad DEPTH_ 8U (0-255).
Histogram: Es la estructura AkHistrogram creada con AkCreateHist.
options: Opciones. Pueden tomar los siguientes valores.

- HIST_ IND: Se calcula el histograma individual de los todos los canales
- HIST_ ALLIN1 : Se calcula el histograma de la suma de los canales RGB

Retorno: AkHistogram con los valores calculados.

Método: AkHist2Akimage

Descripción: AkHist2Akimage permite visualizar los datos del AkHistogram en una estructura AImage representados mediante barras o puntos.
Parámetros:
AkHist: Estructura AkHistogram.
channels: Selecciona los canales a mostrar, puede tomar los siguientes valores:

- HIST_ CHANNEL: Si el AkHistogram solo un canal (Por ser escala de grises o sumatoria de RGB)
- HIST_ RED : Muestra el canal rojo
- HIST_ GREEN : Muestra el canal verde
- HIST_ BLUE : Muestra el canal azul

width:Ancho deseado de la estructura AImage
height: Alto deseado de la estructura Akimage
fill: true/false. True: pinta las barras. False: muestra sólo los máximos.
color:Arreglo de 3 elementos, definen el color de las barras del histograma en modelo RGB.
Retorno:Estructura AImage

Método: AkDFT

Descripción: AkDFT permite calcular la transformación de Fourier de una estructura AImage. Para que AkDFT pueda calcularse, la estructura, debe corresponder a una imagen cuadrada de un sólo canal cuyo ancho y alto debe ser algún valor de potencia de dos. En caso de no cumplir esta condición puede completarse con ceros con AkPaddingZero. También puede utilizarse el método AkDFTPadded que realiza autimáticamente el padding permitiendo trabajar con una imagen de cualquier

tamaño.
Parámetros:
InAkimage: Es una estructura AImage cuadrada con dimensión de potencia de dos.
options: Opciones de transformación. Puede tomar los siguientes valores:

- DXT_ FORWARD: Transformación hacia adelante.
- DXT_ INVERSE: Transformación hacia atras
- DXT_ SCALE: Resultado escalado por 1/NN, donde NN es el número de elementos de la imagen.
- DXT_ ROWS: transformada DFT 1D por filas

swap: true / false. Espectro resultante centrado.
Retorno:Estructura AImage con 3 canales y una profundidad de 32 bits de punto flotante. AkDFT devuelve en el canal R (rojo) la parte real y en el canal G (verde) la parte imaginaria.

Método: AkGetOptimalDFTSize

Descripción: AkGetOptimalDFTSize calcula el valor de potencia de dos superior más cercano. Este método puede utilizarse junto con AkPaddingZero para obtener una estructura adecuada para el método AkDFT.
Parámetros:
Adimension: Valor a partir del cual se calcula el próximo entero potencia de dos.
Retorno:Valor.

Método: AkPaddingZero

Descripción: AkPaddingZero completa con ceros la estructura hasta alcanzar el dimensión especificada por parámetro.
Parámetros:
InAkimage: Es una estructura AImage de entrada.
NewSize: Nuevo valor para la dimensión de la imagen.
Retorno:Estructura AImage.

Método: AkFrequencyFilter

Descripción: AkFrequencyFilter permite realizar un filtrado en frecuencia por medio de la multiplicación del espectro de la imagen con un kernel.
Parámetros:
InAkimage: Estructura AImage correspondiente a una imagen cuadrada con dimensión de potencia de 2 de un único canal.
Kernel: Arreglo unidimensional. Matriz de filtrado 2D cuadrada cuyas filas se en-

cuentran ubicadas consecutivamente conformando un arreglo unidimensional.
swap: true/false. Centrar kernel.
Retorno: Arreglo de dos elementos, el primer elemento es una estructura AImage que contiene la parte real de la transformación, el segundo elemento contiene una estructura AImage que contiene la parte imaginaria de la transformación.

Método: AkFilter2D

Descripción: AkFilter2D permite realizar una filtrado en el dominio espacial mediante una correlación con el kernel pasado por parámetro. soporta AROI
Parámetros:
InAkimage: Estructura AImage de entrada al método.
Kernel: Arreglo unidimensional. Matriz de correlación 2D cuadrada cuyas filas se encuentran ubicadas consecutivamente conformando un arreglo unidimensional.
Anchor: Ancla que fija el pivote del kernel de correlación. Se expresa en un arreglo 1D de dos elementos que definen la coordenada pivote de la matriz.
Retorno: Estructura AImage

Método: AkNonLinealFilter

Descripción: AkNonLinealFilter permite realizar filtrados no lineales sobre una estructura AImage. soporta AROI.
Parámetros:
InAkimage: Estructura AImage de entrada al método.
MaskWidth : Ancho de la máscara de filtrado (supone ser una máscara cuadrada)
Anchor: Ancla que fija el pivote del kernel de correlación. Se expresa en un arreglo 1D de dos elementos que definen la coordenada pivote de la matriz.
TypeOfFilter : Tipo de filtrado. Puede tomar los siguientes valores

- MAXFILTER: Filtro de máxima.
- MINFILTER: Filtro de mínima.
- MODEFILTER: Filtro de Moda.
- MEDIANFILTER: Filtro de mediana

Retorno: Estructura AImage

Método: AkDFTPadded

Descripción: AkDFTPadded permite calcular la transformada de Fourier de una estrucura AImage de cualquier dimensión. En el caso de imágenes donde el tamaño no es potencia de 2 se utiliza relleno de ceros. La inversa retorna la imagen original

con el relleno de ceros.
Parámetros:
InAkimage: Estructura AImage de entrada al método.
options: Opciones de transformación. Puede tomar los siguientes valores:

- DXT_ FORWARD: Transformación hacia adelante.
- DXT_ INVERSE: Transformación hacia atras
- DXT_ SCALE: Resultado escalado por 1/NN, donde NN es el número de elementos de la imagen.
- DXT_ ROWS: transformada DFT 1D por filas

swap: true / false. Espectro resultante centrado.
Retorno:Estructura AImage con 3 canales y una profundidad de 32 bits de punto flotante. AkDFT devuelve en el canal R (rojo) la parte real y en el canal G (verde) la parte imaginaria.

Método: AkErode

Descripción: AkErode permite calcular la erosión sobre una estructura AImage con un elemento estructurante pasado por parámetro. En caso de ser una estructura RGB, se utiliza el mismo elemento sobre todos los canales.
Parámetros:
InAkimage: Estructura AImage de entrada al método.
StrucEle: Es un elemento estructurante cuadrado expresado en un arreglo unidimensional.
Anchor: Ancla que fija el pivote del kernel. Se expresa en un arreglo ID de dos elementos que definen la coordenada pivote de la matriz.
Retorno:Estructura AImage.

Método: AkDilate

Descripción: AkDilate permite calcular la dilación sobre una estructura AImage con un elemento estructurante pasado por parámetro. En caso de ser una estructura RGB, se utiliza el mismo elemento sobre todos los canales.
Parámetros:
InAkimage: Estructura AImage de entrada al método.
StrucEle: Es un elemento estructurante cuadrado expresado en un arreglo unidimensional.
Anchor: Ancla que fija el pivote del kernel. Se expresa en un arreglo ID de dos elementos que definen la coordenada pivote de la matriz.
Retorno:Estructura AImage.

Método: AkLUT

Descripción: AkLUT permite aplicar una transformación por tabla (LUT: Lookup table) a una estructura AImage pasada por parámetro de profundidad DEPTH_ 8U. Si la estructura tiene más de un canal la misma LUT será aplicada en cada uno de ellos. soporta AROI.
Parámetros:
InAkimage: Estructura AImage de entrada al método.
Lut: Arreglo de 256 elementos que corresponden a la tabla de mapeo. El dominio de la tabla es de 0 - 255, en caso de encontrarse fuera de este rango puede escalarse habilitando la bandera Scaled.
Scaled: true/false. Normaliza el arreglo Lut entre el rango 0 - 255.
Retorno:Estructura AImage.

Método: AkGetSize

Descripción: AkGetSize retorna un arreglo de dos elementos con el ancho y alto de la estructura AImage pasada por parámetro.
Parámetros:
InAkimage: Estructura AImage de entrada al método.
Retorno:Estructura AImage.

Método: AkSplit

Descripción: AkSplit permite individualizar una estructura AImage en los diferentes canales que lo componen.
Parámetros:
InAkimage: Estructura AImage de entrada al método.
Retorno:Arreglo de 4 arreglos unidimensionales que corresponden a los valores individuales de los canales en el orden RGBA

Método: AkMerge

Descripción: AkMerge arma una estructura AImage a partir de la mezcla de los canales pasados por parámetro de forma independiente.
Parámetros:
Channel0: Arreglo unidimensional con el contenido específico del canal 0 (rojo). Si es 0 y se completa con ceros.
Channel1: Arreglo unidimensional con el contenido específico del canal 1 (verde). Si es 0 y se completa con ceros.
Channel2: Arreglo unidimensional con el contenido específico del canal 2 (azul). Si es 0 y se completa con ceros.
Channel3: Arreglo unidimensional con el contenido específico del canal 3 (alfa). Si

es 0 y se completa con el valor 255.
OutputAkimage: Estructura Akimage destino donde se arma la estructura resultante. Este parámetro se pasa por referencia.
Retorno:Estructura AImage.

Método: AkCrop

Descripción: AkCrop permite obtener una subimagen de una estructura AImage. El recorte corresponde a la ROI definida en la estructura.
Parámetros:
InAkimage: Estructura AImage de entrada con una ROI definida.
Retorno:Estructura AImage correspondiente a la ROI de la imagen pasada por parámetro.

Método: AkPow

Descripción: AkPow permite elevar cada uno de los elementos de la estructura AImage a un exponente pasado por parámetro.
Parámetros:
InAkimage: Estructura AImage.
Exponent: Exponente a utilizar.
Retorno:Estructura AImage.

Método: AkConvertScale

Descripción: AkConvertScale permite cambiar la precisión de los datos de una estructura Akimage.
Parámetros:
InAkimage: Estructura AImage.
newDepth : Valor de profundidad deseado para la estructura AImage de salida. Este puede ser::

- DEPTH_ 8U: 1 byte sin signo (0, 255). Profundidad por defecto del HTML CANVAS
- DEPTH_ 8S: 1 byte con signo (-127, 128)
- DEPTH_ 16S: 2 bytes con signo
- DEPTH_ 32S: 4 bytes sin signo
- DEPTH_ 32F: 4 bytes flotantes
- DEPTH_ 64F: 8 bytes flotantes

scale:true/false. Si los valores deben ser normalizados al nuevo rango de profundidad.

Retorno:Estructura AImage.

Método: AkCvtColor

Descripción: AkCvtColor permite modificar el modelo de color de una estructura AImage. La estructura AImage de entrada debe tener una profundidad de DEPTH_ 8U.
Parámetros:
InAkimage: Estructura AImage.
ColorCode: Código de color para realizar la conversión. Puede ser:

- RGB2RGBA: Modelo Rojo Verde Azul a Rojo Verde Azul Alfa
- RGB2GRAY: Modelo Rojo Verde Azul a Escala de grises
- RGB2HSV: Modelo Rojo Verde Azul a Tono Saturación Valor
- RGBA2RGB: Modelo Rojo Verde Azul Alfa a Rojo Verde Azul
- RGBA2GRAY: Modelo Rojo Verde Azul Alfa a Escala de grises
- GRAY2RGB: Modelo de escala de grises a Rojo Verde Azul
- GRAY2RGBA: Modelo de escala de grises a Rojo Verde Azul Alfa
- HSV2RGB: Modelo Tono Saturación y Valor a Rojo Verde Azul

Retorno: Estructura AImage.

Método: AkLoadOnCanvas

Descripción: Permite visualizar una estructura AImage en un objeto HTML Canvas. El método respeta los canales definidos previamente en la estructura, ignorando los valores de los canales no considerados. El tamaño del lienzo HTML queda definido por la dimensión de los atributos width y height de la estructura, ignorando aquellos definidos en le objeto HTML CANVAS.
Parámetros:
InAkimage: Estructura AImage.
CANVASReference: es una referencia a un objeto HTML CANVAS.
Retorno:No tiene retorno.

Método: AkAddWeighted

Descripción: AkAddWeighted permite realizar la suma ponderada de dos estructuras AImage. Ambas estructuras deben poseer el mismo valor de profundidad. soporta

AROI
Parámetros:
InAkimage_ 1: Estructura AImage correspondiente al primer sumando.
Weigth_ 1: Peso correspondiente al primer sumando.
InAkimage_ 2: Estructura AImage correspondiente al segundo sumando.
Weigth_ 2: Peso correspondiente al segundo sumando.
Const: (Opcional)Desplazamiento en la suma.
Retorno:Estructura AImage.

Método: AkGet

Descripción: AkGet permite obtener el valor de un elemento particular de una estructura de imagen.
Parámetros:
InAkimage: Estructura AImage de entrada.
x: Valor de la coordenada x.
y: Valor de la coordenada y.
ch: Valor del canal.
Retorno: Valor del elemento.

Método: AkSet

Descripción: AkSet permite definir el valor de un elemento particular de la estructura AImage.
Parámetros:
InAkimage: Estructura AImage de entrada.
x: Valor de la coordenada x.
y: Valor de la coordenada y.
ch: Valor del canal.
val: Valor que para definir en el pixel.

Método: AkClone

Descripción: AkClone permite obtener una copia de una estructura AImage a partir de otra.
Parámetros:
InAkimage: Estructura AImage de entrada.
Retorno:Estructura AImage.

Método: AkResize

Descripción: AkResize permite modificar las dimensiones de una estructura AI-

mage. La estructura debe tener una profundidad DEPTH_ 8U.
Parámetros:
InAkimage: Estructura AImage de entrada.
Width: Ancho nuevo.
Height: Alto nuevo.
Retorno:Estructura AImage.

CAPITULO 4

Experimentos y resultados

En este capítulo se detallan las pruebas llevadas a cabo en el presente proyecto. En primer término se hace una comparación de resultados con otras dos bibliotecas tomadas como referencia, OpenCV y Matlab[1]. A continuación se presentan los resultados de los análisis de portabilidad y compatibilidad de la biblioteca en los navegadores más utilizados. Finalmente se analiza el rendimiento de la biblioteca en un procesamiento en tiempo real, tomando como fuente de procesado la captura de video de la cámara web.

Para evitar redundancia y presentar más comodidad a la lectura, de ahora en adelante a la biblioteca desarrollada se le llamará Akimage.

4.1. Evaluación subjetiva de resultados

La finalidad de este experimento es la de obtener una evaluación subjetiva entre Akimage y dos de las bibliotecas más utilizadas para el procesamiento de imágenes: OpenCV y Matlab. Como ya se mencionó, OpenCV es una bilbioteca desarrollada por Intel y permite trabajar con los lenguajes C, C++ y Python, además permite exportar sus resultados al sistema operativo móvil, Android. Matlab es una herramienta de software matemático que integra un entorno de desarrollo con un lenguaje de *scripting* propio. Se propone como prueba una comparación subjetiva de resultados, ya que por la naturaleza misma de las herramientas evaluadas es incongruente tomar una medida objetiva como la comparación de tiempos de ejecución o medidas cuantitativas de resultados. Desde el punto de vista de los tiempos de ejecución, OpenCV es una biblioteca compilada y posee tiempos de procesamiento muy diferentes a Matlab y Akimage. Por su lado, el compilador de JavaScript varía de navegador a navegador, siendo imposible tomar una medida relativa que sirva

[1] http://www.mathworks.com/products/matlab/

Figura 4.1: Imagen de Lenna

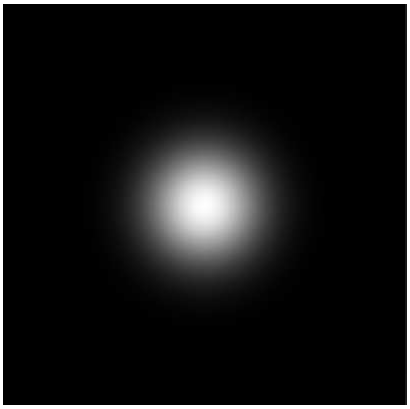


Figura 4.2: Magnitud de filtro gaussiano

como punto de comparación. Por último, la precisión de los datos para visualizar resultados en HTML es fija, por lo que no es posible utilizar medidores cuantitativos.

4.1.1. Definición de las pruebas

Para realizar la comparación se utilizaron tres procesamientos. Primero una conversión de modelo de color, de RGB a HSI, visualizando y comparando los 3 canales resultantes. Luego un filtrado en dominio espacial por convolución, con una máscara pasa altos. Por último, un filtrado pasa bajos en frecuencia con una máscara gaussiana. La imagen escogida para realizar los precesamientos es la conocida "Lenna" (Figura 4.1), con un tamaño de 256 x 256 píxeles, representada en el modelo RGB. Para la tercer prueba se utilizó un filtro gaussiano 2D de 256 elementos con $\mu = 0$ y $\sigma = 0{,}7$ (Figura 4.2).

Comparativa: Conversión de modelo de color RGB a HSI

Para esta prueba se hizo una conversión del modelo de color, de RGB a HSI y se guardaron independientemente las imágenes resultantes de cada canal. Los resultados obtenidos para la componente de Tono se ven en la Figura 4.3, siendo la Figura 4.3(a) el resultado obtenido con Akimage, la Figura 4.3(b) el resultado obtenido con OpenCV y la Figura 4.3(c) el resultado obtenido con MatLab.

(a) Akimage (b) OpenCV (c) Matlab

Figura 4.3: Representación de la componente de Tono, para Akimage, OpenCV y Matlab.

Los resultados obtenidos para la componente de Saturación se aprecian en la Figura 4.4.

(a) Akimage (b) OpenCV (c) Matlab

Figura 4.4: Representación de la componente de Saturación, para Akimage, OpenCV y Matlab.

Finalmente, los resultados obtenidos para la componente de Intensidad se aprecian en la Figura 4.5.

Esta comparación tiene como objetivo evaluar la fidelidad del método de conversión de modelos de color. Los resultados de la prueba no muestras diferencias sustanciales en lo que respecta a los resultados de la transformación de modelo de color para las componentes de Saturación e Intensidad. En cuanto a la componente de Tono, cabe mencionar que OpenCV trabaja con un menor rango de valores que las otras dos herramientas comparadas, esto reduce la resolución del color y

(a) Akimage (b) OpenCV (c) Matlab

Figura 4.5: Representación de la componente de Intensidad, para Akimage, OpenCV y Matlab.

puede producir algunos resultados diversos[21]. En una conclusión general, se puede mencionar que los resultados obtenidos con las 3 herramientas son visualmente semejantes.

Comparativa: Pasa altos en dominio espacial

Para esta prueba se hizo un filtrado en dominio espacial mediante la convolución de la Figura 4.1 con la máscara pasa altos de la ecuación 4.1.

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{4.1}$$

Esta prueba tiene como objetivo evaluar la fidelidad del método de convolución. Los resultados del procesamiento se pueden ver en la Figura 4.6, siendo la Figura 4.6(a) el resultado obtenido con Akimage, la Figura 4.6(b) el resultado obtenido con OpenCV y la Figura 4.6(c) el resultado obtenido con MatLab. Visualmente las imágenes no presentan diferencias significativas.

(a) Resultado Akimage (b) Resultado OpenCV (c) Resultado MatLab

Figura 4.6: Comparativa de filtrado pasa altos con las diferentes bibliotecas

Comparativa: Pasa bajos en el dominio frecuencial

Para esta prueba se realizó una conversión a escala de grises de la Figura 4.1, posteriormente se le aplicó una transformación frecuencial, seguido de una multiplicación punto a punto en el dominio de las frecuencias con el filtro de la Figura 4.2, finalmente se aplicó una transformación frecuencial inversa para expresar la imagen en dominio espacial. Los resultados de este procesamiento se ven en la Figura 4.7, siendo la Figura 4.7(a) el resultado obtenido Akimage, en la Figura 4.7(b) el resultado obtenido con OpenCV y en la Figura 4.7(c) el resultado obtenido MatLab.

(a) Resultado Akimage (b) Resultado OpenCV (c) Resultado MatLab

Figura 4.7: Comparativa de filtrado pasa bajos en el dominio frecuencial

Esta prueba busca evaluar la fidelidad del método de transformación frecuencial. Los resultados arrojaron imágenes visualmente similares entre las herramientas, lo cual concluye que desde los resultados, la biblioteca Akimage tiene un funcionamiento similar al de sus pares.

El objetivo es la de evaluar el funcionamiento de 3 métodos que fueron implementados en este proyecto y que están involucrados en una gran cantidad de algoritmos de procesamientos. Para todos los experimentos se realizó un normalizado en la imagen de salida previa a la visualización, si bien los algoritmos arrojan valores similares, gran parte de las diferencias entre una herramienta y otra se hallan en los criterios utilizados para visualizar. En lo que a Akimage respecta, la visualización debe hacerse en un formato de 4 canales con una precisión de 8 bits por canal (0 - 255). Matlab visualiza independientemente del rango obtenido en el procesamiento (esto le garantiza mayor precisión) y OpenCV mantiene el rango estático dependiendo la precisión de la estructura imagen que esté utilizando, esto puede producir recorte de datos y distorsiones. La normalización realizada permite reducir diferencias visuales y tener un mejor criterio de comparación. Bajo estas condiciones se puede concluir que las pruebas resultaron satisfactorias obteniendo resultados visual y numéricamente similares.

4.2. Comparativa de portabilidad

La portabilidad es una característica básica en las aplicaciones web, ya que son independientes del sistema operativo donde se esté ejecutando el navegador. Sin embargo, los distintos navegadores poseen sus propios interpretes de código JavaScript y en ocasiones estos presentan incompatibilidades entre los mismos métodos o funciones. Akimage fue desarrollada respetando los estándares de la ECMAScript, tratando de alcanzar la máxima compatibilidad con los navegadores. Esta prueba propone evaluar esa portabilidad, comprobando la compatibilidad en distintos navegadores, tanto de versiones de escritorio como versiones portátiles.

4.2.1. Definición de las pruebas

Para la prueba se planteó un algoritmo que involucra lectura y escritura de una imagen en un elemento HTML-CANVAS. El acceso a los datos del elemento HTML-CANVAS es la operación crítica para el funcionamiento de la biblioteca, el resto de las funcionalidades dependen de la información extraída de ese elemento.

Para evaluar los navegadores se utilizó http://www.browserstack.com/, un sitio que permite correr una página web en distintos navegadores y en distintas versiones. Desde allí se puede corroborar el funcionamiento del sitio testeado.

4.2.2. Resultados de las pruebas

En la Tabla 4.1 se ven los resultados de compatibilidad que arrojaron las pruebas sobre el sitio de testeo para navegadores de escritorio. Posteriormente, en la Tabla 4.2 se detallan los resultados de las pruebas realizadas para navegadores de dispositivos móviles[2].

Tabla 4.1: Versiones de compatibilidad mínima para navegadores de escritorio

	Chrome	Firefox	I.E.	Opera	Safari
Versión mínima compatible	4.0	3.6	9.0	9.0	3.1
Versión actual	37.0	32.0	11.0	24	7.0

[2]Valores medidos al 22 de septiembre del 2014

Tabla 4.2: Versiones de compatibilidad mínima para navegadores de dispositivos móviles

	Chrome	Firefox	Opera	Safari
Versión mínima compatible	4.0	3.0	10.0	3.2
Versión actual	35.0	30.0	22.0	7.0

4.2.3. Discusiones y conclusiones

A partir de las pruebas realizadas se concluye que la biblioteca cuenta con una muy buena portabilidad, sus elementos básicos son soportados por los principales navegadores, incluso en versiones, al día de hoy antiguas. A su vez, cabe destacar la buena compatibilidad con navegadores de dispositivos móviles, ya que, como se mencionó a lo largo del informe, son los principales objetivos hacia los cuales es dirigido el presente proyecto.

4.3. Procesamiento de video

En esta prueba se evalúa el rendimiento en el procesado de video de la biblioteca. Para ello se procesaron fotogramas extraídos de la captura de la cámara web cada cierto intervalo de tiempo, registrando intervalos de uso de procesador y el retardo entre la entrada original de video y la salida procesada.

4.3.1. Definición de las pruebas

Para la prueba se procesó cada fotograma muestreado, sin aprovechar ningún tipo de redundancia temporal, lo que implica que cada fotograma fue procesado por completo.

La configuración del hardware usado corresponde a un Procesador Intel Core i5-2430M con CPU 2.40GHz. La frecuencia de muestreo utilizada fue de 30 frames por segundo. Las resolución de la entrada de video de la cámara web es de 640 x 480 píxeles. Esta entrada es escalada a un CANVAS de 256 x 166 (respetando la relación ancho - alto). La entrada de la cámara se puede ver en la parte superior de las capturas de pantalla de las pruebas. Para realizar los procesamientos, esta entrada es escalada y rasterizada en un CANVAS de 256 x 256 píxeles, con el fin de tener una imagen que cumpla con las condiciones necesarias para aplicar una transformación frecuencial. La imagen procesada se ubica en la parte inferior en las capturas de pantalla de las pruebas.

(a) Detección de bordes de fotograma completo

(b) Detección de bordes de fotograma de una región de interés

Figura 4.8: Procesamiento de detección de bordes en tiempo real.

4.3.2. Resultados de las pruebas

Todos los procesamientos, salvo el que involucra una transformación frecuencial, fueron testeados con el fotograma completo y con una ROI.

Detección de bordes con Sobel y aplicación de umbral

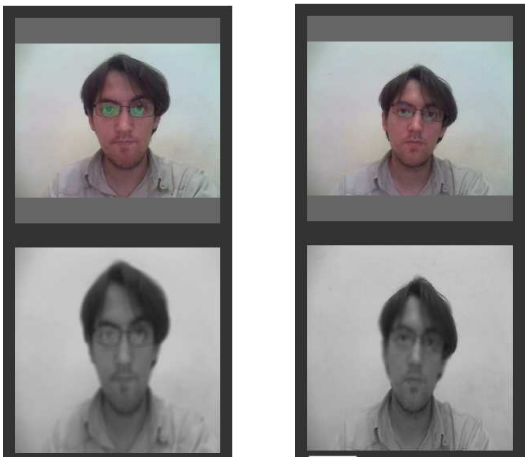
Este procesamiento involucra dos barridas de convolución, inicialmente con una máscara de detección de bordes verticales y posteriormente con una máscara de detección de bordes horizontales. Finalmente se adicionan ambos resultados y se aplica un umbral que permite binarizar y extraer los bordes reconocidos en la imagen.

En la Figura 4.8(a) se observa la captura de pantalla de la imagen procesada en fotograma completo con el procesamiento descrito y en la Figura 4.8(b) el mismo procesamiento para una región de interés de 100 píxeles de alto por 100 píxeles de ancho en el centro de la imagen.

Filtrado blurring mediante aplicación de máscara de promediado

Este procesamiento involucra una barrida de la operación de convolución con la máscara de promediado:

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}.$$



(a) Filtrado de blurring de fotograma completo

(b) Filtrado blurring de una región de interés

Figura 4.9: Filtrado blurring con máscara de promediado.

En la Figura 4.9(a) se observa la captura de pantalla de la imagen procesada en frame completo con el procesamiento nombrado. En la Figura 4.9(b) el mismo procesamiento para una región de interés de 100 pixeles de alto por 100 pixeles de ancho en el centro de la imagen.

Cuantizado de dos niveles

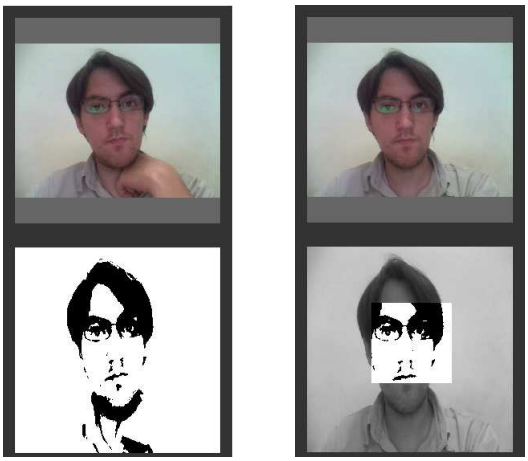
Este procesamiento realiza una transformación punto a punto de binarización. El resultado es un cuantizado de dos niveles, con un umbral de corte elegido para la prueba.

En la Figura 4.10(a) se observa la captura de pantalla de la imagen procesada en frame completo y en la Figura 4.10(b) el mismo procesamiento para una región de interés de 100 pixeles de alto por 100 pixeles de ancho en la región central.

Filtrado pasa bajos en dominio frecuencial

Este procesamiento obtiene la transformada discreta de Fourier de la imagen, seguido de una multiplicación punto a punto con el filtro pasa bajos de la figura 4.2. Finalmente una transformación frecuencial inversa retorna la imagen al dominio espacial. En la Figura 4.11 se observa la captura de pantalla de la salida de la cámara procesada.

Para realizar los experimentos se diseñó una aplicación simple que permite efectuar pruebas de distintos procesamientos y controlar algunas variables como los números de fotogramas por segundo, seleccionar una región de interés, fijar los um-



(a) Binarizado de fotograma completo

(b) Binarizado de una región de interés

Figura 4.10: Procesamiento de cuantizado en dos niveles.

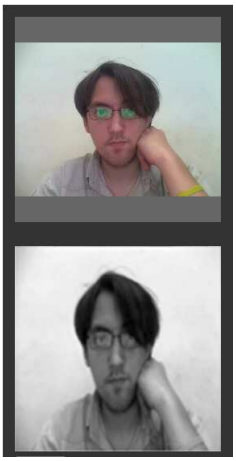


Figura 4.11: Filtrado en frecuencia con un filtro pasa bajos.

brales de corte en los procesamientos que lo requieran y comprobar el valor de un pixel específico. Esta aplicación se puede encontrar en http://akimagejs.com/prueba. En la Figura 4.12 se observa una captura de pantalla de la aplicación online funcionando con algunas de las opciones disponibles para la prueba de la biblioteca.

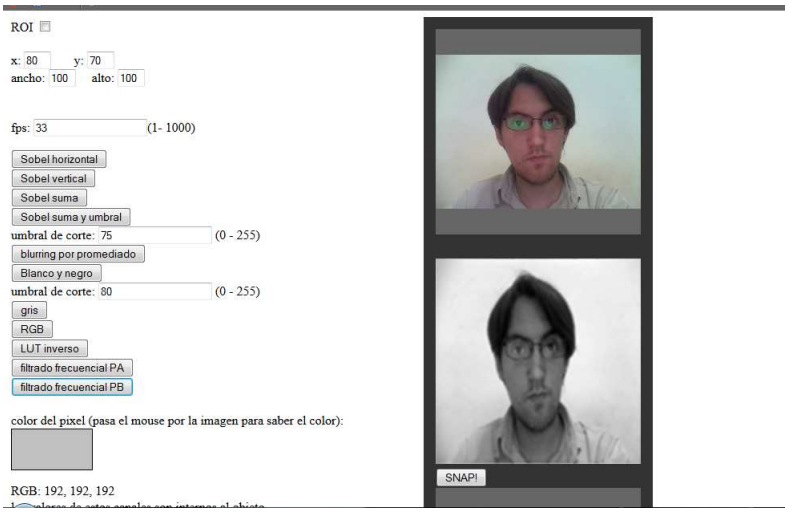


Figura 4.12: Captura de la interfaz para pruebas online.

4.3.3. Discusiones y conclusiones

Para medir el rendimiento de los procesamientos se evaluaron los tiempos de procesamiento y su respuesta a la entrada de la cámara web. En la Tabla 4.3 se observan los tiempos de procesamiento de los distintos experimentos realizados, con frame completo así como también los frame con una región de interés particular. Como se puede apreciar, el proceso más costoso es aquel que involucra una transformación frecuencial, donde se tiene una demora cercana a medio segundo. Éstos valores son aceptables para una aplicación de este tipo, destacando que los procesamientos no involucran el uso de la red ni de los recursos de un servidor. Los datos mostrados en la tabla son el resultado de las pruebas para 30 frames por segundos (FPS), pero no se registró sobrecarga en el uso de memoria al aumentar los FPS debido a la gestión de procesos de la tecnología, que descarta acciones si aún no ha desocupado el uso de la memoria. Esto permite generalizar los usos para distintos dispositivos, sin la necesidad de exigir una capacidad mínima de procesamiento.

Tabla 4.3: Tiempos en milisegundos de los distintos procesos evaluados en las pruebas

Procesamiento	Frame completo	Frame con ROI
Detección de bordes con umbral	280	93
Filtro blurring por correlación	145	55
Cuantización	35	25
Pasa bajos en frecuencia	440	-

CAPITULO 5

Conclusiones y trabajos futuros

En este último capítulo se describen las conclusiones finales obtenidas a partir de las pruebas desarrolladas previamente, además se presentan una serie de propuestas de trabajos futuros que permitirían mejorar el alcance y rendimiento del proyecto.

5.1. Conclusiones finales

En el presente proyecto se planteó el diseño estructural y la implementación de los elementos básicos de una biblioteca para el procesamiento de imágenes, desarrollada íntegramente en tecnologías web estándares del lado cliente del modelo Cliente - Servidor, función actualmente cubierta sólo por herramientas del lado Servidor. Se hizo hincapié en la importancia de la utilización de una tecnología estándar del lado cliente que, como se demostró, puede procesar con la misma funcionalidad en navegadores de diversos dispositivos sin que esto afecte significativamente el resultado. Se nombraron también las principales características de la tecnología. Se planteó la utilización de un diseño basado en un paradigma de Orientación a Datos, que es más versátil y eficiente que el comúnmente usado paradigma de Orientación a Objetos. También se propuso el empleo de una interfaz de métodos basada en la biblioteca OpenCV, al ser esta un referente para el PDI y contar con un gran número de usuarios y desarrollos.

Finalmente, luego de las pruebas realizadas, se comprobó que la biblioteca produce resultados satisfactorios que coinciden con los obtenidos en las mismas pruebas por sus pares de escritorio. También se verificó que el proyecto es compatible con los navegadores más utilizados tantos en sus versiones de escritorio como móviles. Esto fue logrado a partir de la utilización de funciones estándar y compatibles con los navegadores testeados. Por último, se probó, con buenos resultados, los procesamientos implementados en la salida de la cámara web, verificando además de compatibilidad y buen rendimiento, a pesar de no haberse implementado ningún tipo de soporte a

formatos de video.

Desde los orígenes, el proyecto se desarrolló como una herramienta de código abierto para ser cedida a la comunidad al fin de mejorar y ampliar su uso. Se diseñó considerando como objetivo obtener una estructura conocida y definitiva. Para acercar el proyecto a la comunidad, se cargó el código del mismo, así como también ejemplos y documentación, en un repositorio de código libre popular, Github. La dirección de la biblioteca en el repositorio es https://github.com/alefortvi/akimagejs. En una planificación a corto plazo se plantea, también, la concreción del sitio web. Bajo esta perspectiva, los aportes futuros que pudieran hacerse al proyecto son innumerables.

5.2. Trabajos futuros

Para aumentar el rendimiento y alcance de la biblioteca, en esta sección, se proponen una serie de mejoras, con una planificación pautada en plazos en base a las necesidades más urgentes.

A corto plazo

- Mejorar el rendimiento de los métodos de procesamiento implementando algoritmos optimizados de cálculo.
- Ampliar las funcionalidades y mejorar el soporte de archivos de video e integración con cámara Web.

A mediano plazo

- Implementar más métodos de procesamiento para enriquecer la potencialidad y alcance de la biblioteca.
- Ampliar el número de ejemplos en la documentación, que integren los procesamientos con la interactividad del navegador.

A largo plazo

- Implementar integración directa con eventos de los navegadores.

BIBLIOGRAFÍA

[1] Cloud Tweaks, *Plugging into the clound*, Accedida, Julio, 10, 2014. `http://www.cloudtweaks.com/2011/07/10-cloud-based-os-operating-systems`

[2] Berson, Alex , *Client/Server Architecture*, Texas, Estados Unidos, Mcgraw-Hill, 1992

[3] Sommerville, Ian, *Software Engineering 9th Edit.*, United States, Pearson Education Limited, 2011

[4] Oriyano, Sean-Philipp , *Client-Side Attacks and Defense*, Massachusetts, Estados Unidos, Syngress, 2012

[5] W3C, *World Wide Web Consortium* , Accedida, Julio, 10, 2014. `http://www.w3.org/`

[6] Goodman, Danny , *JavaScript Bible 7th Edition*, Indianapolis, Estados Unidos,Wiley Publishing, 2010

[7] W3C HTML Canvas context refference, *World Wide Web Consortium* , Accedida, Agosto, 12, 2014. `http://www.w3.org/TR/2dcontext/`

[8] Meerman, David, *The New Rules of Marketing and PR: How to Use Social Media, Online Video, Mobile Applications, Blogs, News Releases, and Viral Marketing to Reach Buyers Directly*, Texas, Estados Unidos, Wiley Publishing, 2013

[9] Reed, T.V., *Digitized Lives: Culture, Power, and Social Change in the Internet Era*, Nueva York, Estados Unidos, Routledge Publishing, 2014

[10] Gutierrez, Sergio, *Integración Social Digital: Social Media Internet*, Mexico D.F., Mexico, Publicaciones Administrativas Contables Juridicas, 2010

[11] Castillo Helgado, Miguel, *Diario de una Pyme en Internet*, Madrid, España, Publicaciones Gráficas Arias Montano, 2013

[12] Larman, Craig, *UML y Patrones*, Edinburgh, England, Pearson Education Limited, 2007

[13] Suarez, Oscar Deniz, *OpenCV Essentials*, Birminghan, Reino Unido, Packt Publishing, 2014

[14] Android, *Android Operative System*, Accedida Diciembre, 22, 2013.
`http://www.android.com/`

[15] Stefanov, Stoyan, *JavaScript Patterns*, United States, O'Reilly Media, 2010

[16] Harmes, Ross and Diaz Dustin, *Pro JavaScript Design Patterns*, United States, Apress, 2008

[17] R.C. González and R.E. Woods., *Digital image processing*, United States, Prentice Hall, 2002

[18] Steve, Fulton and Jeff, Fulton, *HTML5 Canvas*, United States, O'Reilly Media, 2011

[19] Steven J. Vaughan-Nichols, *Will HTML 5 Restandardize the Web?*, en *Computer IEEE* Vol 43, Issue 4, pp. 13 - 15, abril 2010

[20] L. Wood, *Programming the Web: the W3C DOM specification*, en *Internet computing IEEE* Vol 3, Issue 1, pp. 48 - 54, agosto 2002

[21] HSV Color Conversion, *HSV Color format in OpenCV*, Accedida, Julio, 22, 2015.
`http://www.shervinemami.info/colorConversion.html`