3. F. Durán Muñoz, J. Troya Castilla, A. Vallecillo Moreno. Desarrollo de software dirigido por modelo. Universitat Oberta de Catalunya (2013).
4. I. Sommerville, Ingeniería de Software, 7ma. edición, Pearson, 2005. ISBN: 84-7829-074-5.
5. L. Dyer, F. Henry, I. Lehmann, G. Lipof, F. Osmani, D.Parrott, W.Peeters, J. Zahn. "Scaling BPM Adoption from Project to Program with IBM Business Process Manager". IBM Business Process Manager, EEUU, 2012.
6. Ryan K. L. Ko. "A Computer Scientist's Introductory Guide to Business Process Management (BPM)", ACM New York, NY, USA. Vol.15, Nº4, 2009.
7. OMG, "Business Process Model and Notation (BPMN)", version 2.0, 2011.
8. B. Silver. "BPMN Method and Style: A Levels-based Methodology for BPM Process Modeling". Cody Press, EEUU, 2009. ISBN-10:0982368100.
9. A. Rodriguez, E. Fernandez, M. Piattini. CIM to PIM Transformation: A Reality. In Research and Practical Issues of Enterprise I.S.. Springer Boston. 2008. ISBN 978-0-387-763.
10. G. Booch, I. Jacobson, J. Rumbaugh. El lenguaje unificado de modelado. Segunda Edición. Pearson.2006. ISBN-13: 9788478290765.
11. OMG Unified Modeling Language Infrastructure. Versión 2.4.1 2011.
12. L. Nahuel, E. Santanera, M. C. Ariste, L. Rocca, R. Giandini. Integración Metodológica para el Desarrollo de Tecnologías Software Dirigidas por Modelos y Basadas en Procesos de Negocio. CIINDET 2014.
13. L. Nahuel, E. Santanera, L. Rocca, C. Ariste, R. Giandini. Aportes de las Tecnologías para Gestión de Procesos de Negocio al Desarrollo de Software Dirigido por Modelos. HCITISI 2013, (ISBN 978.88.96.471.25.8).
14. R. Giandini , I. Martinez, L. Mendez, L. Nahuel, J. Perelli, M. Pérsico. Integración de modelos BPMN en ambientes MDA. CACIC-WIS 2012.
15. N. Santos Blasi, M. Pérsico, J. Perelli, I. Martinez Asturdillo, L. Mendez. Desarrollo de Prototipo CASE para Transformación de Modelos en contexto MDD aplicado a Modelos BPMN. JEI 2012.
16. I. Martinez Asturdillo, L. Mendez, J. Perelli, M. Pérsico, N. Santos Blasi, R. Giandini, L. Nahuel. Una aproximación a la generación automática de código en un contexto MDD sobre modelos BPMN. EST 2012 – 41° JAIIO 2012.
17. ATL Lenguaje de Transformación ATLAS https://eclipse.org/atl/.
18. L. Rocca, M. Caputti, I. Zugnoni. Implementando Transformación de Modelos utilizando MOSKitt Tool en adhesión al Paradigma MDD. CONAIISI 2014.
19. MOF Meta-Object Facility. URL: http://www.omg.org/mof/
20. EMF Eclipse Modeling Framework. URL: https://www.eclipse.org/modeling/emf/
21. R. Giandini, G. Pérez, C. Pons. Un lenguaje de Transformación específico para Modelos de Proceso del Negocio. CLEI 2010. Asunción, Paraguay
22. Informe técnico "Especificación de la Transformación de Proceso BPD en BPMN a Diagrama de Actividades UML" PID MAPS 2015. URL: http://maps.frlp.utn.edu.ar/
23. Eclipse Modelling Tool JUNO. URL: http://eclipse.org/juno/ - Fecha de último acceso: Abril/2015
24. ATL SDK. URL: http://eclipse.org/mmt/downloads/?project=atl - Fecha de ultimo acceso: Abril/2015

Desarrollo de una biblioteca para el procesamiento de imágenes utilizando tecnologías web

Alejandro Fort Villa, César Martínez, Enrique Albornoz

sinc(i) Instituto de Investigación en Señales, Sistemas e Inteligencia Computacional - UNL/CONICET,
Facultad de Ingeniería y Ciencias Hídricas,
Universidad Nacional del Litoral. Santa Fe, Argentina
alefortvi@fich.unl.edu.ar,
{cmartinez,emalbornoz}@sinc.unl.edu.ar

Resumen: En la última década se ha desarrollado fuertemente el entorno web. Actualmente, se dispone de una gran cantidad de servicios y herramientas a través de internet. El procesamiento digital de imágenes (PDI) ha adquirido un rol fundamental en este ámbito, dada la importancia que tiene la información multimedial. Sin embargo, no se han desarrollado herramientas web para PDI que realicen el procesamiento en el lado cliente del modelo Cliente-Servidor. En este trabajo se presenta una biblioteca de PDI desarrollada con tecnologías web actuales, basada en la interfaz de la biblioteca OpenCV. Para su diseño se consideró como fundamental su funcionamiento del lado cliente y compatibilidad multiplataforma. Entre los métodos implementados se encuentran: manejo de tipos de datos, transformaciones de modelos de color, filtrado espacial, filtrado frecuencial y operaciones morfológicas, entre otros. La biblioteca fue diseñada modularmente bajo el patrón de diseño Singleton, y con el paradigma de orientación a datos.
Palabras claves: HTML5, JavaScript, OpenCV, Procesamiento de Imágenes

1. Introducción

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). Hoy en día, un gran número de aplicaciones e incluso Sistemas Operativos, residen 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

[1] Aplicaciones que se controlan y ejecutan como entidad independiente.

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), sistemas de posicionamiento, 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 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. Además, es necesario considerar que este esquema le quita autonomía a los clientes y el intercambio de grandes cantidades de información (imágenes, videos, etc.) entre cliente-servidor no siempre es factible, sobre todo utilizando redes de telefonía (3G o anterior). En particular para los dispositivos móviles, la tendencia es que las aplicaciones trabajen de la forma más independientemente posible del servidor (por ejemplo: Facebook messenger, Gmail apps, Yahoo, etc.). A pesar de las desventajas mencionadas, aquellas aplicaciones que implementan procesamientos de imágenes deben optar por tecnologías del lado servidor, pues son las únicas que brindan herramientas para tales operaciones. Por otra parte, las herramientas del lado cliente permiten interactividad y operatividad en tiempo real.

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 [7,19].

En este trabajo se presenta el diseño y desarrollo de una biblioteca para el procesamiento de imágenes con tecnologías web actuales y estándar del lado cliente, que puede ser utilizada en multiplataforma y múltiples navegadores compatibles con HTML5.

La organización del trabajo se presenta de la siguiente manera: La Sección 2 describe el diseño y desarrollo de la biblioteca propuesta. La Sección 3 presenta los experimentos realizados a partir de fotogramas extraídos del flujo de video de una cámara web. Se presenta también una comparativa de portabilidad en diversos navegadores. Finalmente, en la Sección 4 se encuentran las conclusiones y trabajos futuros.

2. Desarrollo de la biblioteca

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 y la relación entre ambos.

2.1. Diseño de la biblioteca

El diseño y la interfaz fueron realizados en base a la biblioteca OpenCV, dado que es una de las herramientas de procesamiento de imágenes más utilizada y

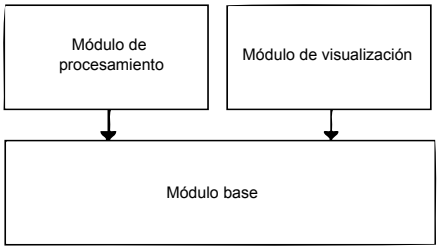


Figura 1. Diagrama de interacción entre módulos.

por lo tanto su estructura ha sido testeada por una vasta comunidad de desarrolladores. Además, se consideró que de esta manera sería factible despertar el interés de los desarrolladores de OpenCV, mientras que las migraciones de proyectos ya consumados podrían realizarse con mínimos ajustes [13,14]. Para el flujo de la información entre los componentes se propuso un diseño ad hoc y los métodos de procesamiento se implementaron, 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 el paradigma de Programación Orientada a Datos.

La estructura de la biblioteca está organizada en 3 módulos principales. El **módulo de procesamiento** está integrado por funciones en las que se implementaron algoritmos de procesamiento específicos del procesamiento digital de imágenes (PDI de ahora en adelante), como ser: convolución de imágenes, filtrado de imágenes, transformada de Fourier, transformaciones de modelo de color. 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 ROI (Region of Interest, 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 1 se puede ver el diagrama de interacción entre módulos. Los módulos de procesamiento y de visualización interactúan exclusivamente con el

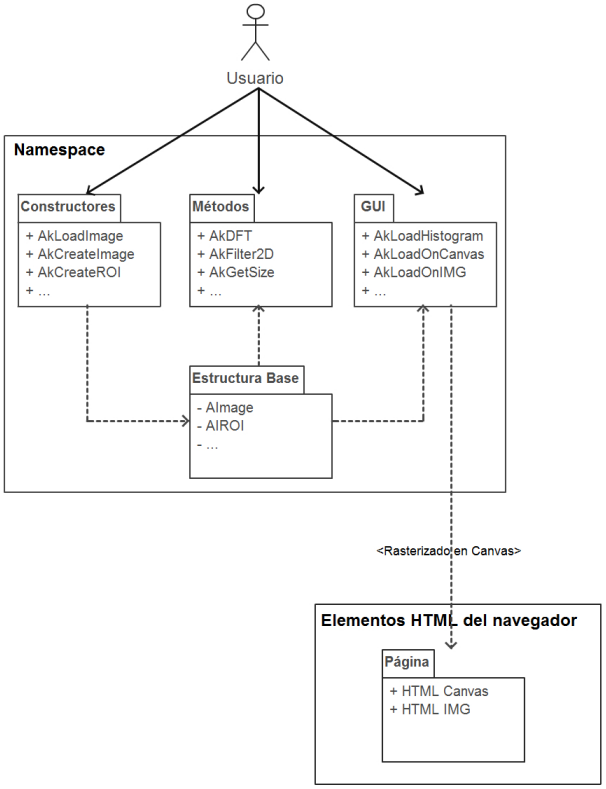


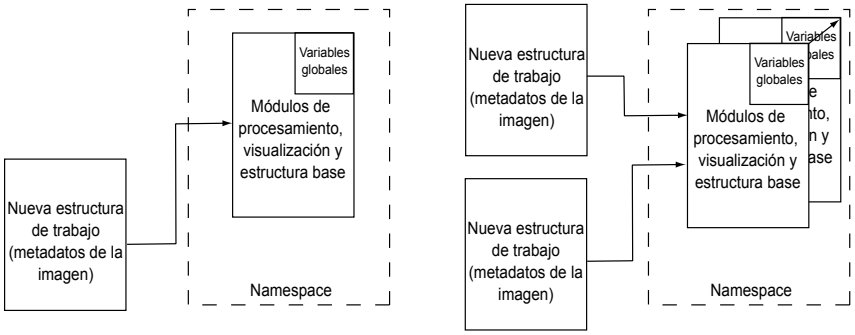
Figura 2. Diagrama UML del diseño general de la biblioteca

módulo base. Las estructuras generadas por el módulo base se utilizan como entrada en los módulos restantes. Las imágenes obtenidas a partir del módulo de procesamiento pueden ser utilizadas nuevamente por éste o bien exportadas por el módulo de visualización.

Para expresar el diseño de la estructura de la biblioteca se utilizó un diagrama de Paquetes UML, ya que si bien un proyecto de estas características carece de una interfaz de interacción y de clases, 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].

En la Figura 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 (Re-

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



(a) Primer instanciación (b) Segunda instanciación

Figura 3. Comportamiento del namespace al instanciarse la estructura base.

gión 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. Los elementos HTML utilizados deben ser declarados en un documento web [19,20].

2.2. Patrón de diseño

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 diseño 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(a) y 3(b) se aprecia el esquema de funcionamiento de la biblioteca con el patrón Singleton. En la Figura 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(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.

3. Experimentos

A continuación se presentan las pruebas llevadas a cabo con la biblioteca: hasta nuestro conocimiento no existen herramientas similares, por lo tanto las comparaciones se realizarán con las bibliotecas de procesamiento más utilizadas (stand-alone). Las comparaciones de tiempos de procesamiento con estas herramientas no es posible debido a que son implementadas de forma optimizada para las diferentes arquitecturas y sistemas operativos sobre los que operan. En primer término se analiza el rendimiento de la biblioteca para procesamientos en tiempo real, considerando el flujo de video de una cámara web estándar. A continuación se presentan los resultados de los análisis de portabilidad y compatibilidad de la biblioteca en los navegadores más utilizados.

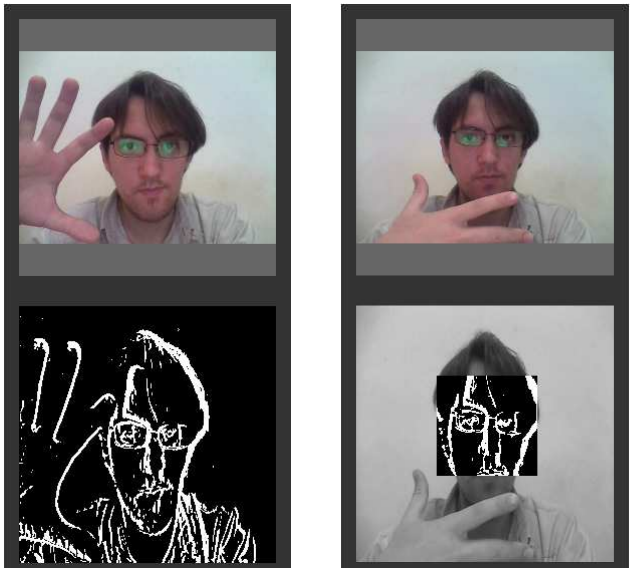
Para evitar redundancia y presentar más comodidad a la lectura, de ahora en adelante a la biblioteca desarrollada se le llamará Akimage.

3.1. Procesamiento en tiempo real

Aquí se evalúa el rendimiento en tiempo real de la biblioteca al procesar fotogramas extraídos del flujo de video de una cámara web, registrando intervalos de uso de procesador y el retardo entre la entrada original de video y la salida del procesamiento. 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 60 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 estas pruebas, la entrada se ha escalado y rasterizado sobre un elemento 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 resultante se ubica en la parte inferior en las capturas de pantalla de las pruebas.

Detección de bordes con Sobel y aplicación de umbral: Este procesamiento involucra dos barridas de correlación, 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.



(a) Detección de bordes de fotograma completo

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

Figura 4. Procesamiento de detección de bordes.

En la Figura 4(a) se observa la captura de pantalla de la imagen resultante en fotograma completo con el procesamiento descrito y en la Figura 4(b) el mismo procesamiento para una región de interés de 100 píxeles de alto por 100 píxeles de ancho en la parte central 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:

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

En la Figura 5(a) se observa la captura de pantalla de la imagen resultante en frame completo con el procesamiento nombrado, en la Figura 5(b) el mismo procesamiento para una región de interés de 100 pixeles de alto por 100 pixeles de ancho en la parte central 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.

(a) Filtrado blurring de fotograma completo

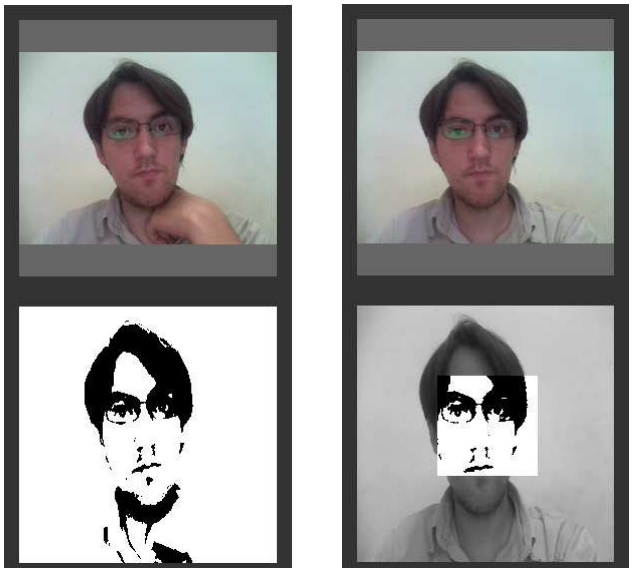
(b) Filtrado blurring de una región de interés

Figura 5. Filtrado blurring con máscara de promediado

En la Figura 6(a) se observa la captura de pantalla de la imagen resultante en frame completo y en la Figura 6(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 gaussiano 2D de 256 elementos con $\mu = 0$ y $\sigma = 0{,}7$ (Figura 8). Finalmente una transformación frecuencial inversa retorna la imagen al dominio espacial. En la Figura 5(a) se observa la captura de pantalla de la imagen resultante de este procesamiento. En la parte superior, la entrada del flujo de video y en la parte inferior la salida procesada.

Interfaz de usuario: Para realizar las pruebas se diseñó una aplicación simple que permite efectuar pruebas de distintos procesamientos y controlar algunas variables como los números de fotogramas de muestreo, seleccionar una región de interés, fijar los umbrales de corte en los procesamientos que lo requieran y comprobar el valor de un pixel específico. En http://akimagejs.com/prueba se puede encontrar esta aplicación. En la Figura 9 se observa una captura de pantalla de la aplicación online funcionando con algunas de las opciones disponibles para la prueba de la biblioteca.



(a) Binarizado de fotograma completo

(b) Binarizado de una región de interés

Figura 6. Procesamiento de cuantizado en 2 niveles.

Al medir el rendimiento de los procesamientos se evaluaron los tiempos de cada proceso y su respuesta a la entrada de tiempo real. En la Tabla 1 se observan los tiempos de procesamiento de los distintos experimentos realizados, con frame completo así como también los frames con una región de interés particular. Como se puede apreciar, el procesamiento 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 60 frames por segundo. Cabe destacar que no se registraron sobrecargas 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.

3.2. Comparativa de portabilidad

Con este análisis evaluamos la portabilidad en distintos navegadores, tanto de versiones de escritorio como versiones portátiles.

Para la prueba se planteó un algoritmo que involucra una lectura y escritura de una imagen al elemento HTML-CANVAS. El acceso a los datos del elemento

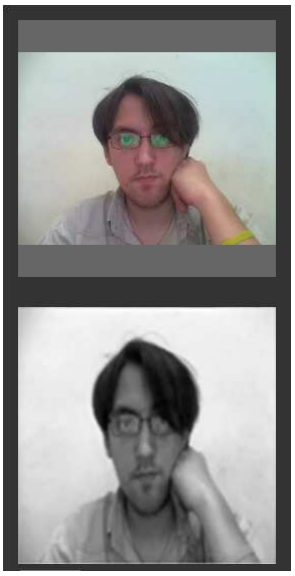


Figura 7. Filtrado pasa bajos en frecuencia.

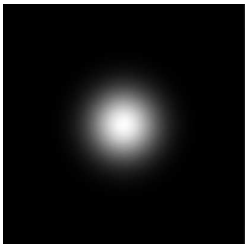


Figura 8. Magnitud de filtro Gaussiano.

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.

Se utilizó el sitio http://www.browserstack.com/ para evaluar los distintos navegadores, éste permite simular la visualización de una página web para un navegador y una versión especificada.

En la Tabla 2 se ven los resultados de compatibilidad que arrojaron las pruebas sobre el sitio de testeo para navegadores de escritorio. Posteriormente, en la Tabla 3, se detallan los resultados de las pruebas realizadas para navegadores de dispositivos móviles.

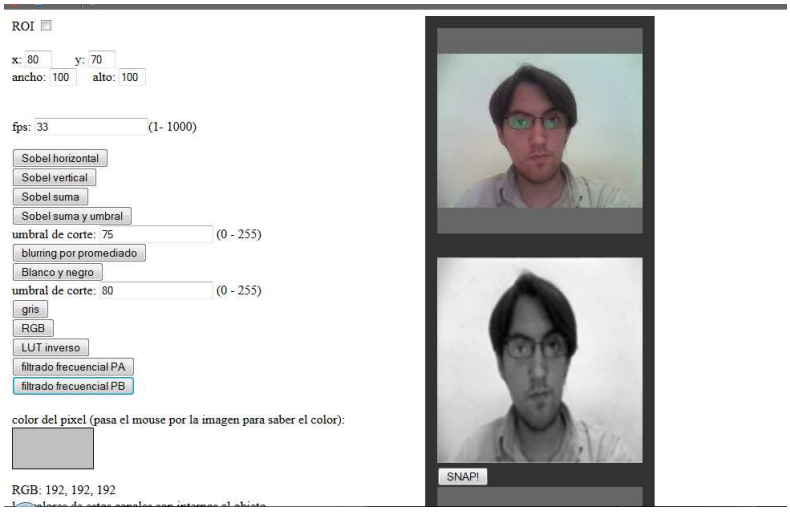


Figura 9. Interfaz de aplicación para pruebas online.

Tabla 1. 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	-

4. Conclusiones

En este trabajo se presentó el diseño e implementación de una biblioteca para el procesamiento de imágenes desarrollada con tecnologías web estándares del lado cliente del modelo Cliente - Servidor. Se planteó el desarrollo utilizando el paradigma de Programación Orientada a Datos. Se propuso el empleo de una interfaz de métodos basada en la biblioteca OpenCV, por ser ésta un referente para el PDI y contar con un gran número de usuarios y desarrollos.

A partir de las pruebas realizadas se concluye que la biblioteca cuenta con una buena portabilidad, sus elementos básicos son soportados por los principales navegadores, incluso en ediciones antiguas, tanto para versiones de escritorio como de dispositivos móviles.

Para aumentar el rendimiento y alcance de la biblioteca, se proponen una serie de mejoras para trabajos futuros:

- Mejorar el rendimiento de los métodos de procesamiento implementando algoritmos optimizados.
- Implementar soporte para archivos de video e integración con cámara Web.
- Incorporar más métodos de procesamiento para enriquecer la potencialidad y alcance de la biblioteca.

Tabla 2. Versiones de compatibilidad mínima para navegadores de escritorio.

	Chrome	Firefox	Internet Explorer	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

Tabla 3. 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

- Implementar integración directa con eventos de los navegadores.

Desde https://github.com/alefortvi/akimagejs puede descargarse el proyecto y su documentación.

Agradecimientos

Los autores agradecen a la *Universidad Nacional del Litoral* (bajo CAI+D 2011 #58-511) y al *Consejo Nacional de Investigaciones Científicas y Técnicas* (CONICET) por su apoyo.

Referencias

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