

## Trabajo final Arquitectura de Computadores/ Interfaces y Arquitectura Hardware

***Student Outcome 06:*** *An ability to develop and conduct appropriate experimentation, analyze and interpret data, and use engineering judgment to draw conclusions.*

### TEMA: APLICACIÓN DE LA CONVOLUCIÓN DE MATRICES AL FILTRADO DE IMÁGENES

#### Introducción

La convolución de matrices tiene muchas e importantes aplicaciones en numerosos campos de la ciencia y la tecnología. Una de las más interesantes es su uso en el filtrado de imágenes, que consiste en la modificación de una imagen de manera que se realcen u oculten algunas de las características de la misma. El presente trabajo final de curso le permitirá desarrollar y conducir un experimento para evaluar el impacto de diferentes versiones de algoritmos con distintos grados de localidad espacial.

Los experimentos desempeñan un papel muy importante en ingeniería porque sirve de apoyo en las fases de validación de un servicio, modelo o desarrollo de un componente software o hardware de un sistema en condiciones más cercanas a los escenarios de trabajo, o cuando se quiere estudiar el comportamiento de un sistema.

A lo largo del curso se estudió la estructura y arquitectura de los computadores, donde el objetivo en sí mismo del curso es evaluar cómo las estructuras hardware y componentes software de un computador afectan el rendimiento y la utilidad de los programas de aplicación, tanto en el procesamiento, acceso y almacenamiento de los datos de los programas que los manipulan. Por tal razón la evaluación y análisis del desempeño es el fin último de un trabajo final.

Un componente importante en la arquitectura de un computador es el sistema de memoria, ya que normalmente se convierte en el cuello de botella de los tiempos de ejecución. Entonces la determinación y medición del tiempo de acceso es crucial para tomar decisiones de configuraciones correctas o adecuadas y determinar y explicar desde la perspectiva del programador, como diferentes programas o algoritmos con la misma complejidad algorítmica pero con diferente grado de aprovechamiento del principio de localidad, podrían tener diferencias significativas a la hora de su ejecución en un computador con una configuración determinada y especialmente en su interacción de con el sistema de memoria.

Para comparar y analizar la eficiencia de los algoritmos y que son escritos en lenguaje de alto nivel, establecer una medida precisa de la eficiencia de un algoritmo no es fácil, ya que, si compara con el tiempo de ejecución, ésta respuesta tiene una gran variabilidad. La complejidad de un algoritmo deberá estar relacionada con el número de operaciones elementales necesarias (asignaciones, comparaciones, sumas, restas, multiplicaciones, divisiones, etc.) para resolver el problema.

El estudio de la estructura y arquitectura hardware de un equipo de cómputo permite comprender cuales son los componentes hardware y elementos software que impactan en el desempeño. Dichos factores van desde el tipo de carga de trabajo o exigencia computacional, tipo de CPU, configuración de la jerarquía de memoria del computador, el sistema operativo, compiladores, lenguajes de programación, los tipos de datos utilizados en el programa, entre otros. De allí la importancia de realizar un diseño experimental bien hecho para establecer los niveles de interacción de los factores y el impacto en la variable respuesta en el que se esté interesado medir.

### Objetivo:

Analizar y emitir un juicio sobre el impacto que tiene en el desempeño en la ejecución de diferentes algoritmos equivalentes, pero con diferentes grados de aprovechamiento del principio de localidad, aplicando para ello la metodología de conducción de experimentos y apoyándose en la teoría de la arquitectura y organización de computadores.

El trabajo consiste en estudiar los efectos que tiene la arquitectura hardware y los elementos software de un computador sobre el desempeño en la ejecución de diferentes versiones de algoritmos equivalentes (con la misma complejidad software). Dichos programas aplican la operación de convolución de una matriz núcleo sobre una imagen de mapas de bits. Especialmente se quiere determinar -mediante un diseño experimental- el punto óptimo desde el punto de vista del desempeño (medido por el tiempo de ejecución).

### Condiciones iniciales del experimento

Para este trabajo tenga en cuenta que debe formar equipos de mínimo 2 y máximo 3 personas.



Cada uno de los integrantes debe disponer de un equipo de cómputo y todos los computadores deben tener el mismo sistema operativo. Cada integrante debe conducir el experimento en su computador.

Como el interés es cuantificar el grado de interacción o dependencia que tiene seis versiones de código (que recorren las matrices de imagen y núcleo con diferentes grados de localidad) con el sistema de memoria, entonces para evaluar dicho impacto del tamaño de las imágenes, debe trabajar imágenes de al menos 12 tamaños diferentes. Para ello debe disponer de imágenes de mapas de bits como el formato .bmp o .tiff, o en su defecto realizar la conversión a mapas de bits con cualquier herramienta de imágenes.

Para facilitar el análisis, se debe convertir o disponer de imágenes a escala de grises de 8 bits de profundidad. Con ello el programa trabajará con matrices de  $n \times n$  con 8 bits para cada píxel.

## Marco conceptual

### Aplicación de la convolución en procesamiento de imágenes.

El procesamiento digital de señales (DSP) ha desempeñado un papel importante en campos como las comunicaciones, visión artificial, teledetección en Cyber-physical systems, pre-diagnóstico en imágenes médicas, edición de fotografías asistida por computador, por nombrar algunos. DSP es la piedra angular de gran parte de la tecnología que usamos hoy en día.

En los sistemas de cómputo modernos, las imágenes se muestran generalmente en una pantalla o monitor con píxeles de imagen discretos, que crean colores con diferentes proporciones de rojo, verde y azul (RGB). En el formato RGB, cada píxel tiene un parámetro para especificar el nivel de cada color, pero para simplificar este trabajo solo tratará con píxeles en blanco y negro (escala de grises). Si bien existen muchos formatos diferentes, UTF-8 (formato de transformación UCS-8-bit) es uno de los más comunes en los sistemas modernos. UTF-8 representa una imagen almacenando la posición y la intensidad de cualquier píxel dado en forma de una matriz 2-D, donde cada elemento de la matriz tiene un valor entero entre 0 y 255 (el rango de valores representables por un número de 8 bits). En UTF-8, el valor 255 es blanco y 0 es negro, con sombras discretas de gris en el medio. Si bien hay mucho más que se puede decir sobre el tema de los formatos de imagen y los esquemas de codificación que tal vez estudiaron en sus cursos de teoría de la información, la profundidad que aquí se cubre es suficiente para comprender los filtros de imagen en escala de grises.

### Filtrado de imágenes

En el procesamiento digital de imágenes existen una gran variedad de procedimientos que permiten, a partir de una imagen, obtener otra modificada (técnicas de filtrado). Se trata de métodos con los que se puede resaltar o suprimir, de forma selectiva, información contenida en una imagen, para destacar algunos elementos o características de ella, o también para ocultar valores anómalos. Los filtros normalmente están basados en la idea de asignar a un píxel el valor en intensidad de color a partir de una ponderación de píxeles cercanos.

### Convolución de matrices

#### Matrices núcleo:

En el procesamiento de imágenes, muchas operaciones de filtrado se aplican a una imagen realizando una operación especial llamada convolución con una matriz llamada kernel. Los núcleos son típicamente matrices cuadradas de  $3 \times 3$ , aunque a veces se usan núcleos de tamaño  $2 \times 2$ ,  $4 \times 4$  y  $5 \times 5$ . Los valores almacenados en el núcleo se relacionan directamente con los resultados de la aplicación del filtro, y los filtros se caracterizan únicamente por su matriz kernel. Por ejemplo, los siguientes núcleos se utilizan para

detectar los bordes verticales y horizontales de una imagen y, cuando se aplican, dan como resultado la imagen que se muestra en la Figura 1.

$$K_h = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, \quad K_v = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$



Imagen original y resultados combinados de aplicación de núcleos de detección de bordes verticales y horizontales.

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

### La convolución

La convolución puede describirse intuitivamente como una función que es la integral o suma de dos funciones componentes, y que mide la cantidad de solapamiento cuando una función se desplaza sobre la otra. Cuando se toma realmente la convolución de dos funciones, una función se voltea con respecto a la variable independiente antes del desplazamiento, y se usa un cambio de variables de  $t$  a  $\tau$  para facilitar la operación de cambio. En una dimensión, las definiciones matemáticas de convolución en tiempo discreto y continuo están indicadas por el operador "\*":

Si  $f$  y  $g$  son funciones en  $t$ , entonces la convolución de  $f$  y  $g$  en un intervalo infinito es una integral dada por:

$$f * g \equiv \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau$$

Entonces los conceptos matemáticos de convolución y la matriz del núcleo se utilizan para aplicar filtros a las imágenes, para realizar funciones como extraer bordes y reducir el ruido no deseado, o aumentar el contraste.

La Figura 2 muestra la convolución de una matriz y una matriz núcleo en una sola coordenada de un pixel específico; la convolución completa se encuentra repitiendo el proceso hasta que la matriz núcleo haya pasado por todos los píxeles posibles de la matriz de origen. En el caso donde las dos matrices son una imagen de origen y un núcleo de filtro, el resultado de convolución es una versión filtrada de la imagen origen. (Por simplicidad asuma ignore los bordes, en esos casos los valores de la matriz resultado toma los mismos que la original).

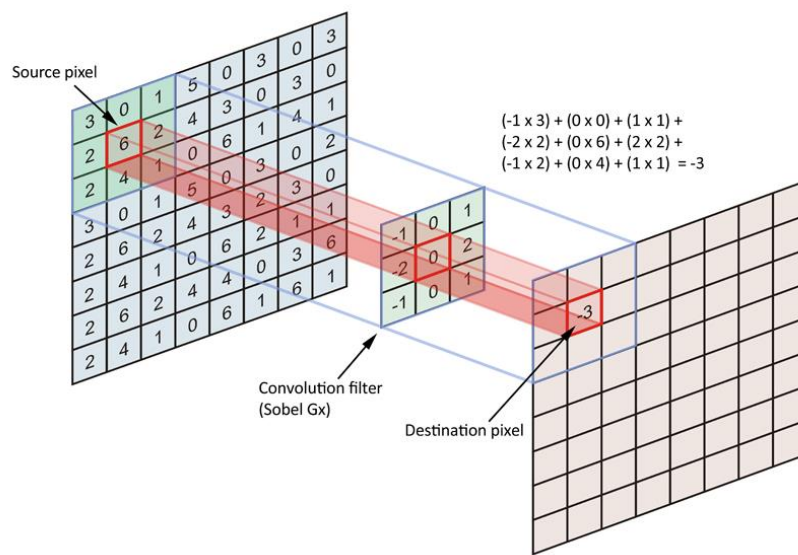
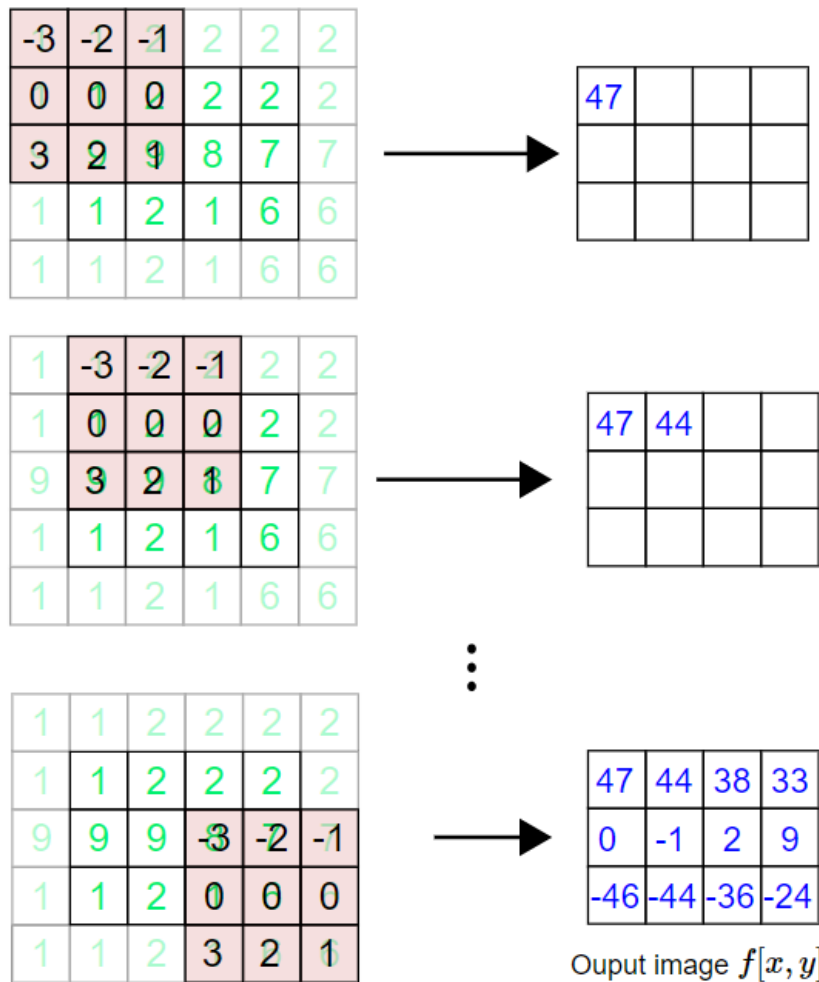


Figura 2: Ejemplo de una sola ubicación en una convolución en 2-D

Tomado de <https://www.gsn-lib.org/docs/nodes/ImageFilterNode.php>

Ventana de convolución

M1	M2	M3
M4	M5	M6
M7	M8	M9

Matriz de convolución

K1	K2	K3
K4	K5	K6
K7	K8	K9

$$PR = M1 * K1 + M2 * K2 + M3 * K3 + M4 * K4 + M5 * K5 + M6 * K6 + M7 * K7 + M8 * K8 + M9 * K9$$

Imagen tomada de: <http://acodigo.blogspot.com.co/2017/05/filtros-de-imagenes-por-convolucion-de.html>

El cálculo del pixel de la imagen resultante se puede realizar aprovechando e implementando varios grados de localidad espacial. A continuación, puede observar un ejemplo de implementación de algoritmo usando una matriz kernel de 3x3. Existe dos *for* anidados externos que recorren los pixels de la imagen indexados por xy. Seguido está otros dos *for* anidados que con i-j indexan la matrix kernel. De esa forma se puede tener 6 versiones de código para realizar la operación de convolución. Una modificación consisten en intercambiar el *for xy*, para recorrer la matriz imagen por filas y otra versión por columnas. Estas dos opciones (xy, yx) se combinan con 3 versiones posibles de indexar la matriz núcleo (ij, ji, unrolling), es decir aprovechando la localidad espacial o no y desenrollando los loops más internos que recorre la matriz núcleo (investigar “[unrolling loop](#)” source: Writing Fast Programs: A Practical Guide for Scientists and Engineers).

```
Im=imread("Ave.bmp");
Im1 = double(rgb2gray(Im));
h = [-2 -1 0; %repujado
     -1 1 1;
     0 1 2];
[r,c] = size(Im1);
```

```
C = zeros(r, c);
for x = 2 : r-1
    for y = 2 : c-1
        for i = 1 : 3
            for j = 1 : 3
                row=x+(i-2);
                col=y+(j-2);
                C(x, y) = (C(x, y) + (Im1(row, col) * h(i, j)));
            end
        end
    end
end
```

} xy , yx

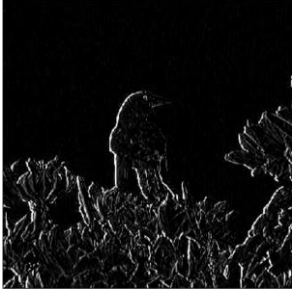



} ij , ji ,unrolling

Ahora, dependiendo de los valores de los elementos, una matriz Kernel producirá diferentes efectos. Para el proyecto usted implementará uno de los tipos de filtros, es decir usando siempre la misma matriz kernel.



Imagen escala de grises de entrada

.Operation	Kernel	Image result	Asignación a grupos de trabajo
Detección de bordes	$\begin{bmatrix} 0 & 1 & 0; \\ 1 & -4 & 1; \\ 0 & 1 & 0; \end{bmatrix}$		G1, G3
	$\begin{bmatrix} -1 & -1 & -1; \\ -1 & 8 & -1; \\ -1 & -1 & -1; \end{bmatrix}$		G2, G4

Sobel	$\begin{bmatrix} -1 & 0 & 1; \\ -2 & 0 & 2; \\ -1 & 0 & 1; \end{bmatrix}$		G5 y G 7
difuminado	$(1/9) \cdot \begin{bmatrix} 1 & 1 & 1; \\ 1 & 1 & 1; \\ 1 & 1 & 1; \end{bmatrix}$		G6 y G8
Repujado	$\begin{bmatrix} -2 & -1 & 0; \\ -1 & 1 & 1; \\ 0 & 1 & 2; \end{bmatrix}$		G9
Tipo Sharpen	$\begin{bmatrix} 1 & -2 & 1; \\ -2 & 5 & -2; \\ 1 & -2 & 1; \end{bmatrix}$		G10

Tomado de: [https://en.wikipedia.org/wiki/Kernel\\_\(image\\_processing\)](https://en.wikipedia.org/wiki/Kernel_(image_processing))

#### Restricciones obligatorias y aclaraciones:

- La variable de salida del experimento será el tiempo de ejecución.
- Se debe utilizar el lenguaje de programación compilado ya sea C++, C o C#.
- En las pruebas preliminares debe evidenciar y comprobar que los 6 algoritmos son totalmente equivalentes.



- d) Recuerde normalizar los datos de tiempo de ejecución, ya que se va manejar diferentes tamaños de matrices. Tome los datos de tiempo total y luego normalice una vez los datos estén exportados a Excel.
- e) Debe escoger cuidadosamente los 12 niveles del factor tamaño de la imagen para que logre un tamizaje de los 3 niveles de memoria cache que tengan los equipos de cómputo que sirven como unidades experimentales.
- f) Puede usar librerías de procesamiento de imágenes solo para facilitar la lectura de las imágenes y conversión a escala de grises y para ver en pantalla las imágenes tanto las originales como el resultado. Pero el proceso de convolución lo debe hacer sin usar los métodos de las librerías si no con el for anidado
- g) Cuando mida el tiempo de procesamiento, no debe incluir las instrucciones de lectura, ni declaración de variables imagen ni visualización en pantalla, solo lo que corresponde al algoritmo, es decir solo los 4 for o 2 para el caso de unrolling.
- h) Los datos recolectados por cada equipo de cómputo, los puede analizar con ANOVA de manera independiente o todos los datos en un solo proyecto de Minitab usando como variable de bloqueo el tipo de computador.
- i) Lea muy bien los pasos de la metodología de diseño de experimentos y tenga en cuenta la rúbrica de evaluación.

En la parte de análisis debe incluir una sección donde explique en detalle y a la luz de los temas del curso, la relación del comportamiento de la memoria y el tiempo de respuesta que describió los resultados del experimento. Tenga en cuenta entre otros conceptos, los principios de localidad, la jerarquía de memoria de cada tipo de procesador y el AMAT. Para efectos de evaluación y calificación se aplicará una rúbrica la cual considera los siguientes aspectos que sigue la mayoría de los pasos de la metodología general de diseño de experimentos:

Aspecto a evaluar	Factor
Diseñar un experimento al seleccionar el tipo más adecuado, de acuerdo a una hipótesis dada, los recursos disponibles y los factores que deben ser medidos y controlados.	Identificación de los factores primarios, niveles y variable respuesta y definición del procedimiento de recolección de datos.
	Identificación de factores secundarios y de ruido y la manera de controlar sus efectos
	Selección del tipo de experimento de acuerdo con la hipótesis / cuestiones por resolver, los factores y las limitaciones existentes
Llevar a cabo un experimento diseñado siguiendo los procedimientos definidos para adquirir datos sobre las variables apropiadas y reportar y consignar adecuadamente los resultados.	La ejecución del experimento siguiendo el procedimiento.
	Reporte de los resultados

Analizar e interpretar los resultados de un experimento para exponer conclusiones que permitan contestadores hipótesis o preguntas del experimento.	Análisis e interpretación de las observaciones y relacionarlas con la teoría del curso.
	Conclusiones apoyadas por los datos reportados

Fecha límite de la entrega del documento: **jueves 10 de noviembre 12m.**

Fechas para la sustentación:

Cada grupo se reservará una hora en una de las tres jornadas posibles

- a) Jueves 10 de junio en la tarde de 14:00 a 18:00
- b) viernes 11 de junio en la mañana de 8:00 a 12:00
  - viernes 11 de junio en la tarde de 14:00 a 18:00.

### Entregables

La tarea en intu será el único medio permitido para entregar el trabajo final.

Por favor suba entregue en una archivo **.ZIP** lo siguiente:

1. Documento metodología, análisis y resultados DoE(.DOCX)
2. Datos resultados (.XLS)
3. Proyecto MiniTab
4. Código fuente y archivos fuente (imágenes)

### Referencias:

[1] 1.Computer Systems: A Programmer's Perspective, Randal E. Bryant, David R. O'Hallaron. Addison Wesley Pub Co Inc; Edición: 0002 (Febrero de 2010).

[2] Memory hierarchy and access time. Tomado de:  
<http://sandsoftwaresound.net/raspberry-pi/raspberry-pi-gen-1/memory-hierarchy/>

[3] Diseño y análisis de experimentos Douglas C. Montgomery

[4] Writing Fast Programs: A Practical Guide for Scientists and Engineers, John Riley.

[5] Aplicación de la convolución de matrices al filtrado de imágenes. F. Gimenez-Palomares, J. A. Monsoriu, E. Alemany-Martínez, Universitat Politècnica de Valencia  
<https://polipapers.upv.es/index.php/MSEL/article/view/4524>  
doi: 10.4995/msel.2016.4524.