

Sistemas Operativos 1/2021

Laboratorio 4

Profesores:

Cristóbal Acosta (cristobal.acosta@usach.cl)

Fernando Rannou (fernando.rannou@usach.cl)

Ayudantes:

Cristóbal Fernández (cristobal.fernandez@usach.cl)

Manuel I. Manríquez (manuel.manriquez.b@usach.cl)

Benjamín Muñoz (benjamin.munoz.t@usach.cl)

I. Objetivos Generales

Este laboratorio tiene como objetivo general la aplicación de los principios de localidad temporal y espacial. Se espera que los estudiantes implementen dichos principios mediante un programa en lenguaje C en un sistema operativo basado en Linux.

II. Objetivos Específicos

1. Estudiar los principios de localidad temporal y espacial
2. Hacer un programa que aplique un filtro de suavizado a una imagen mediante el lenguaje de programación C
3. Escribir las imagenes resultantes en archivos
4. Medir tiempos de ejecución
5. Practicar uso de `Makefile` para compilación de programas

III. Conceptos

III.A. Localidad

III.A.1. Localidad temporal

Cuando se accede a una posición de memoria, es probable que esa dirección de memoria sea accedida nuevamente en un lapso de tiempo corto, es decir, una dirección de memoria que fue utilizada recientemente puede volver a ser referenciada próximamente.

III.A.2. Localidad espacial

Este principio indica que si una dirección de memoria es referenciada, es probable que se referencie una dirección cercana o contigua próximamente. En otras palabras, al referenciar un dato, es posible y bastante probable que se referencie a un dato vecino.

III.B. Filtro Suavizado

Técnica que sirve para suavizar los bordes de un imagen, reducir el ruido o disminuir los cambios de intensidad en la imagen. En este caso se hará uso del filtro media, en el cual se obtiene el pixel de salida obteniendo la media aritmética de su vecindario.

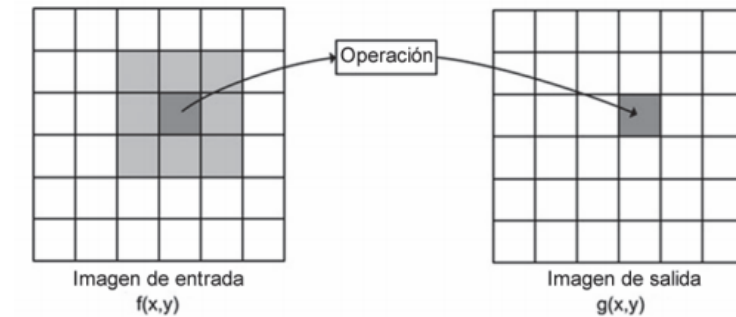


Figure 1. Ejemplo de obtención de pixel de salida



Figure 2. Ejemplo de imagen pasada por filtro de suavizado

IV. Enunciado

Para este laboratorio se pide crear un programa en C que permita aplicar el filtro de suavizado aplicando los principios de localidad temporal y espacial, utilizando los siguientes métodos:

IV.A. Método 1

En este método, se debe aplicar el filtro de suavizado recorriendo la imagen por FILAS. Una vez recorrida una fila, la siguiente fila debe partir desde la última columna de la fila anterior, por ejemplo, para una imagen de $M \times N$ se recorre la primera fila hasta la columna N , y en la segunda fila debe partir iterando desde la columna N hasta la primera columna, mientras que en la tercera partira nuevamente desde la columna 1, y así sucesivamente hasta recorrer toda la imagen. Una vez terminado el procesamiento, la imagen se debe escribir un archivo resultante.

130	160	70	91	101	24
45	23	58	205	109	199
200	42	13	5	100	116
280	183	128	123	8	223
245	87	26	118	133	101
71	222	145	167	182	198

Figure 3. Ejemplo de recorrido por filas

IV.B. Método 2

Este método busca aplicar el método suavizado con los principios de localidad temporal. Para este caso, se debe aplicar el suavizado por COLUMNAS, donde una vez terminada una columna, se recorre la siguiente columna desde su primera fila. Al igual que en el método anterior, se pide escribir la imagen resultante en un archivo.

IV.C. Pruebas

Con el fin de analizar el comportamiento de ambos métodos para el efecto de suavizado, se pide calcular los tiempos de ejecución para cada caso, lo cual se puede realizar con la función `clock()` u otra que se estime pertinente. Para esto debe probar las distintas imágenes, y hacer variadas pruebas con cada uno de los métodos, medir sus tiempos de ejecución y entregar datos como tablas y gráficos en un informe adjunto.

V. Parámetros de Entrada

Para este trabajo se debe entregar al menos un archivo llamado `lab4.c` y un *Makefile* que entregue un archivo ejecutable final llamado `lab4`. Cada imagen estará en formato raw (`float`). La ejecución del programa tendrá los siguientes parámetros que deben ser procesados por `getopt()`:

- `-I filename` : especifica el nombre de la imagen de entrada
- `-F filename` : especifica el nombre de la imagen final con el método 1
- `-S filename` : especifica el nombre de la imagen final con el método 2
- `-M número` : especifica el número de filas de la imagen
- `-N número` : especifica el número de columnas de la imagen
- `-b`: bandera que indica si se entregan resultados por consola. Si se utiliza este flag debe imprimir los tiempos de ejecución para cada método.

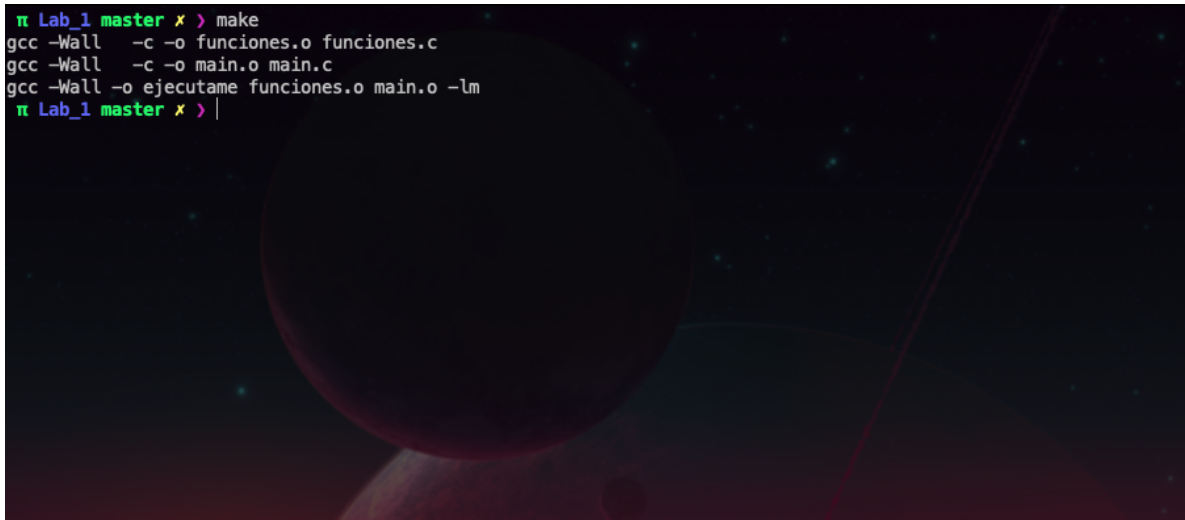
Por ejemplo:

```
./lab4 -I imagen1.raw -F imagen1F.raw -S imagen1S.raw -M 256 -N 256 -b
```

VI. Entregables

El laboratorio es individual o en parejas y se descontará 1 punto por día de atraso. Cabe destacar que no entregar un laboratorio implica la reprobación de la asignatura. Finalmente, se debe subir en un archivo comprimido **formato zip** a Uvirtual los siguientes entregables:

- **Informe:** Un informe en formato artículo (dos columnas) con no más de dos planas de extensión. Debe explicar brevemente su desarrollo, pruebas con sus resultados, un análisis y conclusiones. Debe ser en formato PDF.
- **Makefile:** Archivo para **make** que compila el programa. El makefile debe compilar cada vez que haya un cambio en el código. Si no hay cambios, el comando **make** no debe crear un nuevo objeto.



```
π Lab_1 master x > make
gcc -Wall -c -o funciones.o funciones.c
gcc -Wall -c -o main.o main.c
gcc -Wall -o ejecutame funciones.o main.o -lm
π Lab_1 master x > |
```

Figure 4. Ejemplo de funcionamiento de un Makefile

- **archivos .c y .h** Se debe tener como mínimo un archivo .c principal llamado lab4.c con el main del programa. Además, deben existir más archivos con el proyecto (*.c, *.h) . Se debe tener mínimo un archivo .h que tenga cabeceras de funciones, estructuras o datos globales. Se deben comentar todas las funciones de la forma:

```
//Entradas: explicar qué se recibe
//Funcionamiento: explicar qué hace
//Salidas: explicar qué se retorna
```

- Se considerará para evaluación las buenas prácticas de programación, como modularidad y nombres de variables representativos.
- Trabajos con códigos que hayan sido copiados de un trabajo de otro grupo serán calificados con la nota mínima.

El archivo comprimido debe llamarse: RUTESTUDIANTE1_RUTESTUDIANTE2.zip

Ejemplo: 19689333k_186593220.zip

NOTA: Si los laboratorios son en parejas, éstas puede ser de distintas secciones, pero se debe avisar previamente vía correo a los ayudantes (Cristóbal Fernández y Benjamín Muñoz).

VII. Visualización de las imágenes

Tanto las imágenes de entrada como de salida estarán en formato binario (**float**). Por lo tanto, no pueden ser visualizadas directamente por los software de imágenes. Luego, usaremos Matlab (disponible

gratuitamente a todos los estudiantes de la Universidad) para visualizar imágenes. El siguiente script `verimagenraw()` realiza el trabajo.

```
function a = verimanageraw('filename', M, N)
    f = fopen(filename, 'r');
    a = fread(f, 'float');
    a = reshape(s, N, M); % si, es N, M, no M, N
    a = a'; % al trasponer queda de MxN
    figure; imagesc(a); axis('square');axis('off');
    colormap(gray)
```

Aunque el script abre una ventana y muestra la imagen, también retorna el arreglo (la imagen) como una matriz, la cual usted podría seguir usando. Por ejemplo, para visualizar con una colormap distinto:

```
a = verimagenraw('imagen_1.raw', 512, 512);
figure; imagesc(a); axis('square');axis('off');
colormap(hot)
```

VIII. Fecha de entrega

Domingo 22 de Agosto 2021, hasta las 23:55 hrs.