DESARROLLO DE SOFTWARE EN ARQUITECTURAS PARALELAS

PRÁCTICA 4: ARISTAS

La práctica se ENTREGARÁ en el directorio:

/home/CUENTA/ARISTAS (CUENTA=cuenta personal).

Los ficheros o directorios que sea necesario crear pero no vayan a ser entregados al profesor deberán crearse directamente en el directorio /home/CUENTA y no dentro del subdirectorio de prácticas ARISTAS.

En esta práctica se desea que se realice la implementación paralela, sobre MPI, de aplicación de una máscara de Laplace para extraer contornos o aristas de una imagen.

Las imágenes con las que trabajar estarán en formato PGM debido a su facilidad de uso. Los archivos PGM son archivos portables que contienen datos de imagen en forma de mapas de escala de grises. Estos archivos pueden guardarse mediante dos representaciones: como texto sin formato (ASCII) o como archivos binarios, indicado en ambos casos en el encabezado del archivo PGM. Si el encabezado incluye la cadena "P2" se trata de texto, mientras que si la cadena es "P5" es una representación binaria. En nuestra práctica usaremos la representación binaria. El encabezado del archivo contiene diversos datos, como la anchura y la altura de la imagen y el número máximo del valor de gris (normalmente 255). La información de la imagen se almacena en un array 2D con distintos números enteros, que representan la escala de gris del píxel correspondiente de la imagen, desde negro (0) hasta blanco (255).

Para almacenar estas imágenes usaremos una estructura en c:

```
typedef   struct {
          int row;          // alto de la imagen
          int col;          // ancho de la imagen
          int max_gray;     // máximo valor de gris
          int **matrix;     // array 2D con los valores de cada píxel
} PGMData;
```

La máscara para extraer contornos o aristas de una imagen corresponde a una pequeña matriz de 3×3 elementos con contenido:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix},$$

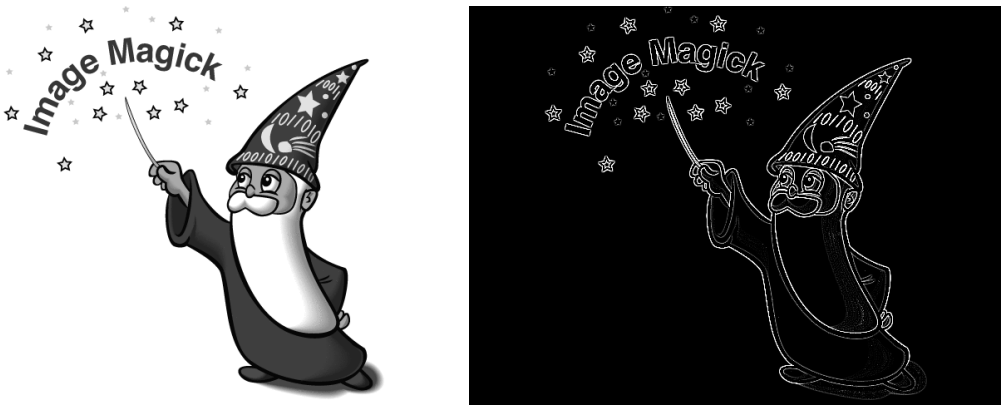
que se aplica a los elementos de la imagen original y su efecto es resaltar los contornos de la imagen. Su aplicación sobre la matriz de píxeles de una imagen consiste en:

$$\begin{aligned} imagen\_edge[i][j] \leftarrow\ & 8 * imagen[i][j] \\ & - imagen[i-1][j-1] - imagen[i-1][j] - imagen[i-1][j+1] \\ & - imagen[i][j-1] - imagen[i][j+1] \\ & - imagen[i+1][j-1] - imagen[i+1][j] - imagen[i+1][j+1] \end{aligned}$$

Existen diferentes estrategias para tratar los píxeles de los bordes (que no tienen 8 vecinos alrededor). Una de las más comunes consiste en dejar a cero (negro) los bordes de la imagen. Otra posibilidad similar, que es la que aplicaremos en esta práctica, consiste en cambiar a 255 (blanco) estos bordes. Con ello podremos detectar ciertos errores en la construcción del algoritmo paralelo.

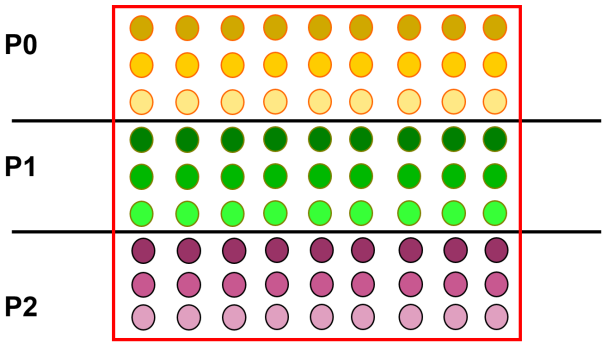
Tras filtrar la imagen con la máscara anterior, los píxeles de valor menor que 0 se convierten al valor mínimo (0, negro), y los de valor mayor que 255, al valor máximo (255, blanco); ver versión secuencial, aristas.c).

Las siguientes imágenes muestran un ejemplo de aplicación a partir de una imagen original:

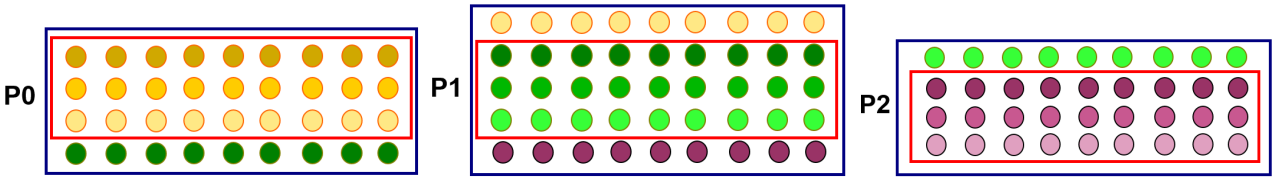


La implementación debe usar un esquema de variables similar a la implementación secuencial suministrada. El proceso número 0 (proceso root) se encargará de leer la imagen original. Adicionalmente, una vez aplicado el filtro de Laplace de forma paralela, será el responsable de guardar la imagen de aristas.

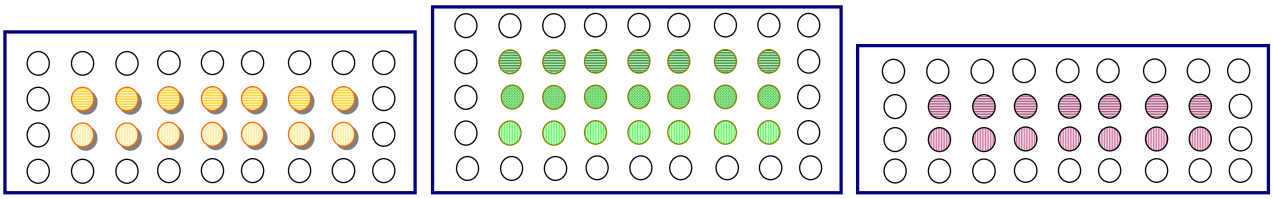
La implementación paralela del algoritmo estará basada en una descomposición unidimensional por bloques de filas consecutivas de la matriz de píxeles de la imagen. Si el número de procesos no divide al número de filas, el exceso de filas se le asociará al proceso número 0 (proceso root) (ver ejemplo psdot.c). Cada proceso será responsable de la actualización de una o más filas adyacentes de la matriz de píxeles. La siguiente imagen muestra la distribución de píxeles de una imagen 9×9 entre 3 procesos:



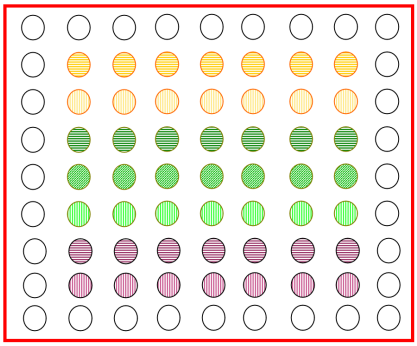
Teniendo en cuenta el esquema de dependencias en el que la actualización de cada píxel se realiza a partir de sus ocho vecinos más próximos, cada proceso necesitará el siguiente almacenamiento en el ejemplo anterior.



A partir de estas imágenes locales, cada proceso aplica la máscara sobre su imagen.



Una vez finalizado, las filas correspondientes de las imágenes locales se envían al proceso root que guardará la imagen de aristas.



Ficheros a entregar:

aristas_mpi.c Contendrá la unidad principal.

aux.c Contiene las funciones auxiliares (ya suministrado con la versión secuencial).

Archivos de imágenes Imágenes usadas como input.

makefile Makefile utilizado para la compilación.

Unidad `aristas_mpi`

- **Lectura de datos**: El proceso 0 se encarga de leer la imagen original en la estructura `img_data` del tipo `PGMData`.
- **Envío y recepción de datos comunes a todos los procesos**: el número de filas y columnas de la imagen, `row` y `col` (`row = img_data.row` y `col = img_data.col`).
- **Cálculo del número de filas asignadas a cada proceso**: Cada proceso calcula el número de filas con las que va a trabajar: `rowlocal`.
- **Envío y recepción de filas de la matriz de píxeles de la imagen**: El proceso 0 envía las filas de `img_data.matrix`. El resto de procesos no es necesario que reciban dentro de una estructura `PGMData`; pueden recibir sus filas en un array 2D de enteros, por ejemplo en el array `matrix`. Notar que además de recibir un total de `rowlocal` filas, cada proceso distinto del 0 debe disponer también de la fila superior y de la fila inferior para poder realizar correctamente el filtro de Laplace (el último proceso solo dispondrá adicionalmente de la fila superior).
- **Aplicación del filtro de Laplace**:
  - El proceso 0 aplica el filtro de Laplace solo a sus filas de la matriz de píxeles de la imagen. El resultado lo almacena en el array `img_edge.matrix` de la estructura `img_edge` del tipo `PGMData`.
  - Los procesos distintos de 0 aplican el filtro de Laplace sobre el array 2D `matrix`. El resultado del filtro lo almacenan en otro array 2D, `matrix_edge`.
- **Envío y recepción de resultados**:
  - El proceso 0 recibe las distintas filas correspondientes a la aplicación del filtro de Laplace por parte del resto de procesos. La recepción se realiza directamente en el array `img_edge.matrix` de la estructura `PGMData`.
  - Los procesos distintos de 0 envían sus filas del array 2D, `matrix_edge`.
- **Escritura de resultados:** El proceso 0 guarda la imagen resultado de la detección de contornos, `img_edge`, en archivo, siguiendo el formato `PGM`. El nombre del archivo debe ser `nombreImagen_edge_paralelo.pgm`, donde `nombreImagen` se refiere al nombre del archivo con la imagen original, `nombreImagen.pgm`.