Tecnológico de Monterrey

Act 2.2

Periodo Semestral Agosto-Diciembre 2022
Fecha de entrega |*Octubre 10, 2022*

Multiprocesadores TE3061.1

Profesor:

Emmanuel Torres Rios

Abril Jimenez Alatriste | *A01732412*

Instituto Tecnológico de Estudios Superiores de Monterrey

Instrucción

Usar dos imágenes diferentes (<700 píxeles por lado y >2000 píxeles por lado) para desenfocar y rotar la imagen. Debes paralelizar la tarea y comparar el tiempo de ejecución con respecto a la tarea realizada sin el uso de hilos. Como resultado de su proceso, debe generar 2 archivos .gif (usando algún servicio en la red), con los archivos generados. Al menos 20 imágenes dentro de cada .gif. deberá presentar un informe con los resultados obtenidos y el código utilizado para las pruebas

Introducción

Al procesar una imagen, lo que nos interesara es identificar los objetos representados en ella para poder realizar un análisis más detallado de estos objetos, por ejemplo podemos contárlos, midiendir sus tamaños, entre otras. Un concepto importante asociado a la identificación de objetos en una imagen es el de los bordes: las líneas que representan una transición de un grupo de píxeles similares en la imagen a otro grupo diferente. Un ejemplo de un borde son los píxeles que representan los límites de un objeto en una imagen, donde termina el fondo de la imagen y comienza el objeto.
Con esta idea endemos a este grupo de pixeles lo conoceremos como kernel.
En pocas palabras, el desenfoque se refiere a la parte de una imagen que está desenfocada.

Marco teórico.

Se realizo un programa el cual, tenia como objetivo realizar el efecto de blurring, o bien de desenfoque y ademas de realizar el “flip-peo” de la misma y finalmente hacer gif, con la pagina web adjuntada.

https://ezgif.com/resize

Resultados:
De efecto de Blurring.
Con un kernel de 3x3, uno de 7x7 y finalmente de 9x9.

toad.bmp Imagen 1.1

toad3x3.bmp Imagen 1.2

toad7x7.bmp Imagen 1.3

toad9x9.bmp Imagen 1.4

Como podemos observar entre mayor kernel mayor sera el fecto de blurring.

Juntaremos la matriz de 7x7 como ejemplo.

```
//Matriz de 7x7
int linea=0;
int cuenta2=0;
int inicioLin=0;
int finLin=ancho-1;
int valor=0;
int a=1;
const double startTime = omp_get_wtime();
#pragma omp parallel
{
  #pragma omp for schedule(dynamic)
    for (int i = 0; i < alto*ancho; i++) {
```

```
if(linea<=3 || ( i==(inicioLin) && i<=(inicioLin+3) ) || ( i>=(finLin-3) && i==(f
  fputc(ptr[i], outputImage);
  fputc(ptr[i], outputImage);
  fputc(ptr[i], outputImage);
}else{

  for(int j = -4; j <= 4 ; j++){
    a = i+(ancho*j);
    valor = valor + ((ptr[a]+ptr[a-1]+ptr[a-2]+ptr[a-3]+ptr[a-4]+ptr[a+1]+ptr[a+2
  }
```

```
&& i<=(inicioLin+3) ) || ( i>=(finLin-3) && i==(finLin) ) || linea>=(alto-4) ){




[a-1]+ptr[a-2]+ptr[a-3]+ptr[a-4]+ptr[a+1]+ptr[a+2]+ptr[a+3]+ptr[a+4])/81);
```

```
            fputc(valor, outputImage);
            fputc(valor, outputImage);
            fputc(valor, outputImage);
            valor=0;

          }
          cuenta2++;
          if(cuenta2==ancho){ //fin de linea
            cuenta2=0;
            linea++;
            inicioLin=linea*ancho;
            finLin=inicioLin+(ancho)-1;
            //printf("linea: %i\n", linea);
          }
        }
    }
```

Conclusión.

En conclusion, podemos concluir como vimos anterior mente entre mayor el kernel mayor el efecto, ahora bien el flippeo no lo pude realizar pues al igual que en el programa anterior se tenia el problema de la duplicacion de la imagen por lo que se concluye que de debera investigar mas o intentar tomando en cuenta el programa anterior.