Proyecto 1: Aplicación para la generación de gráficos y texto

Arquitectura de Computadores 1

Luis Chavarría Zamora / Jeferson González Gómez

Noemí Vargas Soto
Ingeniería en Computadores
Instituto Tecnológico de Costa Rica
Cartago, Costa Rica
mimi2002@estudiantec.cr

***Abstract*—In this project, an application was created that implements bilinear interpolation on a 400x400 grayscale image. The implementation was done in Python, where everything related to the image was handled: quadrant division, selection of the quadrant to interpolate, ordering of the interpolated values, and displaying the resulting interpolated image. The interpolation algorithm was implemented in x86 Assembler, which receives a 2x2 matrix corresponding to a submatrix of the original matrix and returns a 4x4 matrix, corresponding to the interpolated matrix. Python is responsible for joining the interpolated submatrices. The connection between both languages was established through subprocess. The results can be observed through an interface in Tkinter Python.**

***Index Terms*—interpolación bilineal, imágenes, ensamblador, python**

I. Introducción

La interpolación se trata del proceso de calcular valores numéricos desconocidos a partir de otros ya conocidos mediante la aplicación de algoritmos concretos. [1]

Por otro lado, se encuentra la interpolación bilineal, la cual es una técnica muy usada para el procesamiento de imágenes, para redimensionar o transformar imágenes a partir de los valores de sus pixéles (valores vecinos).Esta es una extensión de la interpolación lineal para interpolar funciones de dos variables (por ejemplo, x, y) en una malla regular de dos dimensiones. La idea principal es realizar una interpolación lineal en una dirección, y después en la otra. Aunque cada uno de estos pasos es lineal, la interpolación en su conjunto no es lineal sino cuadrática. [2]

En este proyecto se implementa un algoritmo para aplicar interpolación bilineal a imágenes. A partir de la imagen escogida, se puede obtener una nueva imagen de tamaño superior a la inicial, rellenando la información con datos obtenidos a partir del algoritmo. [1]

II. Desarrollo

A. Listado de requerimentos del sistema

A partir del planteamiento del problema, se identifican los siguiente requerimentos:

Requerimientos funcionales:

- Manipulación y procesamiento de imágenes usando Python.
- División de imágenes en cuadrantes.
- Ejecución del código ensamblador desde Python.
- Aplicación del algoritmo de interpolación bilineal en ensamblador.
- Reconstrucción y visualización de la imagen interpolada en una interfaz.
- La manipulación de archivos debe ser en formato .img.

Requerimientos no funcionales:

- Facilidad de uso a través de una interfaz gráfica.
- Compatibilidad con sistemas Linux.

La implementación de este algoritmo de interpolación para imágenes en Python en conjunto con Ensamblador, permite obtener el máximo provecho de ambos lenguajes. La facilidad de Python para manejar y acomodar valores, además de la manipulación de la imagen y, en un lenguaje de bajo nivel como lo es ensambalador, donde se implementó el algoritmo de interpolación, el cual podría ser difícil de encontrar pero, puede resultar más eficiente, especialmente cuando hay limitanciones en los dispositivos.

B. Opciones de solución al problema

Se plantearon dos soluciones posibles para la integración de Python y Ensamblador con el objetivo de interpolar imágenes en escala de grises utilizando submatrices 2x2. Desde el principio se mantuvo el mismo enfoque para el desarrollo del proyecto, lo que ha cambiado es en algunos aspectos de formato.

Algunos aspectos a tomar en cuenta:

- La interfaz es la misma para ambas versiones. Es una interfaz desarrollada con Tkinter, Python donde se muestra una imagen 400x400 en escala de grises. Dicha imagen tiene 16 cuadrantes de 100x100 divididos con líneas negras. Después de seleccionar el cuadrante, este se marca con líneas rosado fuerte, además de que se muestra la imagen del cuadrante por aparte.

Fig. 1. Interfaz de la aplicación

- Después de elegir el cuadrante, se crea un archivo .img con los 10000 pixeles del cuadrante, un valor por línea. Este documento es tomado por otra sección de código que genera todas las submatrices 2x2 posibles dentro del cuadrante y las guarda cada una en un archivo .img diferente, con el mismo formato de un valor por línea. Estos documentos son reescritos en valores binarios para que sean procesados por el código de Ensamblador y generar la salida respectiva. Esta salida se termina de procesar en Python, ordenar valores y demás, para mostrar la imagen en la interfaz.

1) Opción de solución 1:: En esta primera versión, la idea era que el agoritmo de ensamblador leyera los 4 valores en formato texto del documento .img. El problema de este enfoque radicó en que no se logró identificar cuántos dígitos tenía el número de esa línea, además, la escritura en el archivo output.img en formato texto, con los valores resultantes de la interpolación, no se guardaban correctamente. La intención de manejar los datos en formato texto era la de simplificar ciertos cálculos, así los datos salían casi que directos para ser graficados, pero resultó ser un enfoque más complejo de programar.

2) Opción de solución 2:: El segundo enfoque de solución, se concentra en lo siguiente: después de generar los documentos de las submatrices se crea una función recursiva, la cuál va agarrando un documento de submatriz, lo convierte a formato binario (formato que recibe el código Ensamblador para este enfoque de solución). Python se encarga de llamar al código Ensamblador, con el archivo correspondiente, este se ejecuta, se genera un archivo de salida en formato binario, que Python agarra, traduce a texto, reacomoda los valores, esto se realiza recursivamente hasta cumplir con todos las submatrices. Despupés de tener todos las submatrices interpoladas, todos estos documentos pasan por un código que genera la imgen interpolada y la muestra.

Todos los archivos generados se guardan en documentos .img.

C. Comparación de las propuestas de solución

La opción 1 requiere lógica extra para la conversión de strings a número, dificultad en el manejo de caracteres, y puede presentar errores de escritura. Pero esta opción presenta la ventaja de que se puede realizar verificación de los datos, y que son legibles.

La opción 2 presenta mayor velocidad de procesamiento, menor complejidad en el manejo de los datos, pero su gran desventaja es que comprobar los valores es mucho más compejo, ya que no se pueden leer directamente y requiere una conversión a formato texto.

D. Opción elegida

La opción 2 fue la elegida por simplicidad en cuanto al manejo de de datos, lectura y escritura en ensamblador. Además de que no se requiere la conversión de valores. También resulta ser más sostenible, mejora el rendimiento y es la opción más viable cuando se tiene recursos limitados.

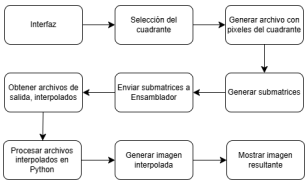

Fig. 2. Diagrama de la Solución 2, solución escogida

III. Conclusiones

La implementación del algoritmo de interpolación bilineal puede ser complicado en un princpio. Se debe tener muy claro el paso paso del proceso para que todo vaya saliendo y funcionando.

El diseño escogido no resulta ser el más eficiente, pero se concluye que funciona de forma correcta, aunque toma algunos segundos extra ejecutarse.

References

[1] EMEZETA., ”Interpolación de imágenes”. [Online]. Available: https://www.emezeta.com/articulos/interpolacion-de-imagenes/.

[2] Wikipedia., ”Interpolación bilineal”. [Online]. Available: https://es.wikipedia.org/wiki/Interpolaci%C3%B3n_bilineal.