Proyecto Individual: Desarrollo de una aplicación para la generación de gráficos y texto

1st Gabriel Guzmán Rojas
Ingeniería en Computadores
Instituto Tecnológico de Costa Rica
Cartago, Costa Rica
gabguzman@estudiantec.cr

I. Listado de requerimientos del sistema

El sistema debe permitir hacer una interpolacion bilineal a un cuadrante de 16 opciones de una imagen 390x390, que esté en la escala de grises. El formato de la imagen es .img , de misma manera, en un .py aparte en la carpeta del programa, se encuentra un convertidor de imagen .Jpeg o .png a .img. El cuadrante seleccionado de la imagen tiene que guardarse de misma manera en formato .img, y este pasaria a ensamblador para ser interpolado. El sistema es un combinacion de lenguaje de alto nivel y bajo nivel, siendo el de alto nivel utilizado unicamente para hacer una interfaz que le permita al usuario seleccionar de una imagen el cuadrante a interpolar, y despues de seleccionado, mostrar el cuadrante seleccionado y el cuadrante ya interpolado, haciendo una comparacion entre ellos.

A. Requerimientos funcionales

Con la anterior explicacion basica de lo que tiene que realizar el programa, se pueden desprender los siguientes requerimientos funcionales:

- Leer y guardar una imagen en formato .img
- La imagen tiene que ser de 390x390
- El usuario tiene la posibilidad de elegir un cuadrante de 16 en que se divide la imagen
- Se tiene que utilizar Ensamblador para interpolar la imagen. Las opciones disponibles son ARM, X86 o RiscV.
- Se tiene que utilizar un lenguaje de alto nivel para la visualizacion de las imagenes.
- El lenguaje de alto nivel se utiliza unicamente para mostrar la imagen de entrada, el cuadrante seleccionado y el cuadrante ya interpolado. El lenguaje de alto nivel no interpola ni hace nada a la imagen.

B. Requerimientos no funcionales

Entre los requerimientos no funcionales del sistema, tenemos que se tiene que trabajar en git, se necesita utilizar Syscall y por lo mismo se tiene que utilizar Linux.

C. Estado del Arte

La interpolación bilineal es un método de remuestreo que escala o manipula imágenes y cuadrículas. En esencia, esta técnica consiste en estimar un nuevo valor de píxel dentro de una matriz utilizando los cuatro valores de píxel más cercanos, directamente adyacentes tanto vertical como horizontalmente. Este método calcula el nuevo valor mediante una interpolación lineal, primero en una dirección y luego en la otra. Al ampliar una imagen, a menudo se encuentran intersecciones en blanco con datos desconocidos. La interpolación bilineal cubre estas lagunas considerando la proximidad de los valores de píxel conocidos alrededor del punto en blanco. La importancia de este metodo o el uso que se le da es para mejorar la calidad de imagen, preservar la calidad de los detalles y por eficencia [1].

Desde una perspectiva académica, la interpolación bilineal también se estudia por su facilidad de implementación matemática y su aplicabilidad directa en arquitecturas de computadores. Implementarla a bajo nivel (por ejemplo, en lenguaje ensamblador) permite desarrollar habilidades clave en el manejo de memoria, control de registros y optimización de algoritmos en contextos donde no se puede usar hardware gráfico especializado. En resumen, la interpolación bilineal representa una solución eficiente y efectiva para generar imágenes más suaves y detalladas, tanto en aplicaciones prácticas como en entornos educativos centrados en arquitectura y procesamiento digital.

D. Estandares y Normas

Para este proyecto se deben respetar una serie de estándares definidos en el enunciado. En primer lugar, el procesamiento de la imagen debe realizarse estrictamente en uno de los lenguajes ensamblador permitidos: ARM, x86 o RISC-V. La imagen utilizada debe tener dimensiones fijas de 390×390 píxeles(se puede hablar con el profesor, pero para efecto de este trabajo se utiliza ese tamaño), y debe estar almacenada en formato .img. Este formato de la imagen representa los valores de cada píxel en una escala de grises de [0-255], donde cada valor ocupa un byte. Asimismo, el cuadrante seleccionado y procesado también debe almacenarse en formato .img antes y después de ser interpolado.

Además, se debe cumplir con las siguientes normas:

Separación clara entre visualización (alto nivel) y procesamiento (bajo nivel).

Uso de solo syscalls para el manejo de archivos.

Correcto uso y desarrollo del repositorio, aplicando buenas practicas.

II. Opciones de solución

A. Basada del Enunciado

La solucion principal que se penso para realizar este problema, y de las cuales se fueron pensando otras soluciones, es la solucion que se nos brindo del enunciado del proyecto mismo, que indica como manejar la matriz y como manejar los valores nuevos. La solucion que se propone en base al enunciado es el obtener un cuadrante de 97x97 de imagen y la imagen que se crea para interpolar va tener un tamaño de 289x289, esto para que todos los pixeles sean cubiertos. Esto se obtuvo en base a la formula de:

$$Tamaño = 97 + (97 - 1) * 2$$

Utilizando buenas practicas, el objetivo es dejar bien planteado el como se aborda el problema en ensamblador, por lo que se tiene como objetivo el separar claramente en el codigo cada parte de la interpolacion, de forma que se trabaje de una forma modular en el mismo archivo, como se ve en el siguiente diagrama.

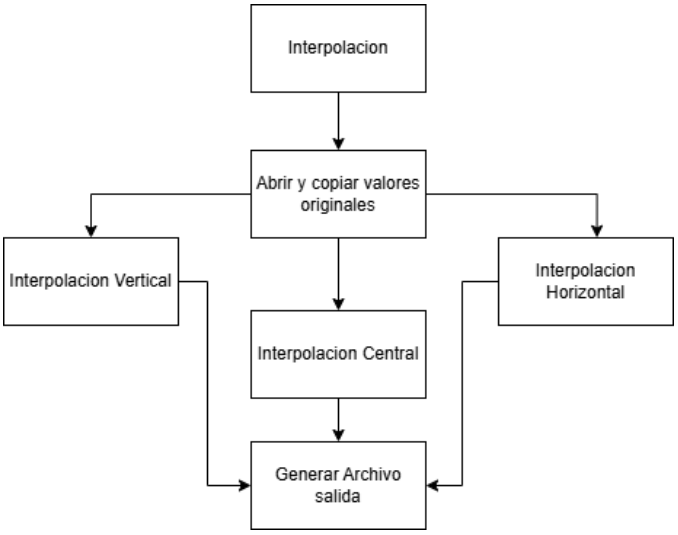

Fig. 1. Diagrama modular

En base a ese tamaño, y siguiendo el enunciado se puede expandir la matriz de forma que pase de la original de 2x2 a 4x4.

A	B
C	D

A	e	f	B
j	X	Y	m
n	W	Z	K
C	g	h	D

Al usar esta expansion me permite utilizar las siguientes formulas para los valores expandidos.

$$e = \frac{2A+B}{3}$$

$$f = \frac{A+2B}{3}$$

Y de misma manera, se utilizan estas mismas para el calculo de las verticales, pero usando los originales correspondientes. La matriz de misma manera se va recorriendo cumpliendo el recorrer 3 espacios, es decir pasar a la siguiente matriz, utilizando la siguiente columna. Como se ve en la sigueinte figura.

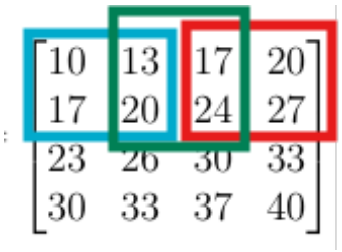

Fig. 2. Recorrido visual matrices

Donde se modifica lo que se presenta en el enunciado, es en el calculo de los valores centrales, ya que se penso que seria mas efectivo utilizar la siguiente formula:

$$X = \frac{2A+B+C+D}{6}$$
$$Y = \frac{A+2B+C+D}{6}$$
$$W = \frac{A+B+2C+D}{6}$$
$$Z = \frac{A+B+C+2D}{6}$$

Siendo esta solucion la que mayor mantiene una calidad visual, una alta precision al estar trabajando con ecuaciones mas exactas y cumple con lo que se propone mayormente en el enunciado.

B. Expansion 2x2 a 3x3

Esta propuesta de solucion parte de la iniciativa de utilizar casi todo lo propuesto en la primera solucion propuesta, que es bastante fiel al documento del proyecto dada por el profesor Luis Chavarria, en el sentido de utilizar las mismas para calcular los valores interpolados y casi el mismo concepto

de recorrido de la matriz, pero la diferencia mayor es que en esta propuesta se penso en utilizar una matriz 3x3, esto para reducir el recorrido que se hace en la matriz, y optimizar el calculo de valores intermedios a calcular unicamente uno cada par de pixeles.

Solucion: Expansión de 2×2 a 3×3

A	B
C	D

A	e	B
g	i	f
C	h	D

Siendo de esta manera que como se aprecia en la imagen, se crearia e, g, f, h, i como valores intermedios de la matriz, que como se ve en la imagen, va diferente a la idea que cada dos pixeles originales, se inserten 2 intermedios.

Una de las principales ventajas de este enfoque es que reduce considerablemente la carga computacional. Al requerir solo un cálculo de interpolación por dirección, disminuye:

- El número total de operaciones aritméticas (particularmente divisiones)
- El uso de registros temporales en ensamblador
- El tamaño total de la imagen interpolada

Otra ventaja si asi se pudiese considerar de esta solución es que permite generar una imagen interpolada de tamaño 193×193 en vez de los 289x289 que generaria la solución 4×4. En esta solucion tambien se tiene que contemplar que hay que seleccionar una de las dos formulas de los intermedios que se conocen y se utilizan en la primera solucion, es decir las que operan con 2/3 o 1/3, o cambiar la formula para el calculo. Por lo que la expansión de 2×2 a 3×3 representa una solución intermedia entre precisión y eficiencia. Ofrece un resultado visual razonablemente bueno, manteniendo las ventajas estructurales de la interpolación bilineal, pero con una implementación más sencilla y ligera.

C. Divisiones Exactas

Otra de los posibles soluciones que se penso es en utilizar diferentes formulas para obtener los valores intermedios y medios, esto debido a que en ensablador y mas precisamente en X86 que es el elegido para la realizacion del sistema, si no se utiliza punto flotante para la divisiones, el sistema suele ”truncar” los valores de la division, y en la formula original, se presentan dos divisiones diferentes, que seria el caso de 1/3 y 2/3, y en lo que se penso fue en reducir la ecuacion a una version mas simplificada a obtener los valores intermedios horizontales y veticales con solo una division para el resultado.

A	B
C	D

A	e	f	B
j	X	Y	m
n	W	Z	K
C	g	h	D

$$e = \frac{A+B}{3}$$

$$f = \frac{A+B}{2}$$

Estas operaciones evitan el uso de multiplicaciones o sumas ponderadas, lo cual reduce el número de instrucciones y, más importante aún, evita divisiones que pueden llegar a ser propensas a error en x86 sin punto flotante.

Por otro lado,como se aprecia en la figura anterior, se genera una matriz 4x4, lo que haria que se tuviesen que calcular dos intermedios cada dos pares, que es lo propuesto en el enunciado del proyecto, pero que se calcularian los valores interpolados de una forma diferente. Esta solucion presenta la ventaja de utilizar casi la misma memoria que consume la imagen original, las operaciones son mas simples por lo que los datos evitarian truncamientos y ademas es la que mejor rendimiento daria.

D. Separacion de Archivos

Otra solucion que se tiene contemplada, pero que no modifica directamente el método de interpolación, sino que plantea una forma alternativa de estructurar el procesamiento que consistiria en dividir el archivo .img del cuadrante completo en bloques más pequeños que pueden ser procesados individualmente. La idea central es segmentar el cuadrante seleccionado en subimágenes de menor tamaño, por ejemplo, bloques de 8×8 o 16×16, que luego se interpolan por separado y se unen al final para formar la imagen completa. Esto tiene diversos motivos de porque se piensa de esta manera, ya que reiterando lo ya mencionado, la interpolacion se mantendria utilizando las mismas formulas de la solucion 1 o la solucion 3, pero el resultado final variaria porque no es un solo .img. Esta propuesta viene pensada desde un punto de vista tecnico, y para no exigir demasiado al sistema, mas que en mi caso se estaria utilizando una maquina virtual de ubuntu, y el recorrido de una matriz gigante consume mucho recurso, por lo que este metodo o solucion permite tener mas control en el consumo de recursos, el manejo de memoria y la carga de registros. De una forma resumida, esta solucion funcionaria de la siguiente manera:

1) Se seleciona el cuadrante a trabajar en python
2) Se separa el cuadrante en diferentes archivos .img. Ejemplo: Bloque1.img

3) Cada uno de estos archivo se pasan a interpolar a ensamblador
4) Una vez interpolados y generados por ensamblador pasan a otro archivo de python
5) El archivo de python unicamente une estos archivos en un solo .img.
6) Este archivo final .img seria el que se mostraria en la interfaz de python.

En este diagrama se puede apreciar mejor la idea.

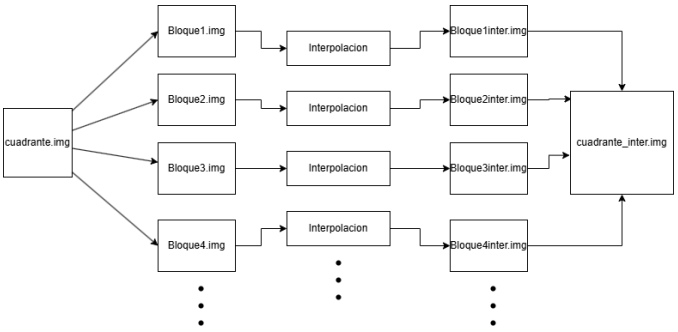

Fig. 3. Enter Caption

E. Comparacion propuestas de solucion

Cada una de las soluciones tiene sus ventajas y sus desventajas por lo que comparando las tres opciones propuestas para implementar el algoritmo de interpolación bilineal, se tienen que tomar en cuenta criterios técnicos y teóricos relevantes como la fidelidad del modelo de forma matematica, la eficiencia computacional en ensamblador, la precisión numérica y el efecto visual sobre la imagen resultante.

La primera solucion que se propuso fue la de interpolación bilineal con expansión 2×2 a 4×4 que sigue fielmente lo que se propone en el enunciado del proyecto, aplicando ponderaciones exactas de 2/3 y 1/3 en los píxeles intermedios horizontales y verticales, y 1/6 combinando los cuatro píxeles A, B, C y D para calcular los valores centrales de la matriz generada. Esta implementación mantiene la coherencia con el modelo teórico de interpolación bilineal 2D, respetando las proporciones y generando transiciones suaves en todas direcciones. Sin embargo, esta opción implica mayor complejidad computacional, mayor presión sobre los registros y uso frecuente de divisiones enteras, pero realmente es la única que garantiza resultados visualmente precisos y matemáticamente consistentes. Basandonos en el contexto del proyecto donde la calidad visual y la adherencia al modelo son prioritarios esta es la solución más adecuada.

La segunda solucion que se propuso que fue la de interpolacion bilineal con expansion 2x2 a 3x3 se basa en un modelo de interpolación mínima, en el cual solo se genera un valor intermedio entre cada par de píxeles, tanto en dirección horizontal como vertical, y un único valor central. Esta solucion permite reducir significativamente el número de cálculos y el uso de memoria, por lo que es altamente eficiente desde la perspectiva de arquitectura de la computadora y la exigencia a la misma. Sin embargo, desde el punto de vista teórico, esta solución no cumple completamente con el modelo de interpolación bilineal, ya que ignora las relaciones diagonales entre píxeles. Visualmente, esto puede generar artefactos como bordes angulosos o transiciones abruptas, y en la parte matematica puede generar resultados que van a afectar directamente a la calidad visual, principalmente la calidad del suavizado.

La tercera solucion propuesta que es la correspondiente a Divisiones simplificadas propone reemplazar las fórmulas bilineales exactas por aproximaciones fácilmente evaluables en x86, como (A + B)/2 o (A + B)/3. Este enfoque va a permitir maximizar la eficiencia, ya que puede evitar divisiones complejas y en su lugar utilizar desplazamientos binarios o sumas simples. Esta opcion no estan exigente al procesador ni a los registros, y el recorrido de memoria es similar al de la primera solucion, pero este enfoque rompe completamente con la fidelidad al modelo bilineal, ya que no considera las posiciones relativas ni los pesos correctos de los píxeles, lo que introduce errores acumulativos debido al uso de mismos valores en otras matrices y distorsiones visuales. Desde una perspectiva teórica, no puede ser considerado como interpolación bilineal propiamente dicha, sino como una interpolación media aproximada.

La cuarta solución propuesta fue la de separacion de archivos que se basa en la separación del archivo.img en múltiples archivos más pequeños, los cuales se interpolan por separado y luego se recombinan para formar la imagen final expandida. Esta solucion no modifica realmente la interpolación propuesta, sino que organiza el proceso de manera modular. El objetivo es dividir para hacer el codigo más manejable, ya que al procesar cada uno de forma independiente mediante el mismo código ensamblador, y al final unirlos, reduce significativamente la presión sobre los registros y simplifica el manejo de índices, ya que se trabaja con estructuras más pequeñas y localizadas. Además, permite reutilizar un mismo algoritmo de interpolación sin necesidad de adaptarlo para matrices grandes, por lo que se facilita el proceso de depuración y hasta pensando en los temas vistos en arquitectura se permitiria paralizar si fuese lo que se estuviese pidiendo. Pero el problema de esta solución es que se requiere que el lenguaje de alto nivel se encargue de dividir y recombinar los bloques, lo cual haria un paso mas en el flujo de trabajo lo que podria llegar a complicarlo, porque pues puede dar problemas el reconomiento de los bloques a la hora de unirlos y tambien puede provocar muchos pixeles vacios a la hora unirlos, lo que daria una mala imagen, pero sigue siendo una solucion sólida, eficiente y escalable que permite mantener una buena calidad visual y que ademas distribuye mejor la carga de trabajo de cada parte.

En términos de rendimiento, las soluciones 2 y 3 ofrecen mayor eficiencia y menor complejidad, pero comprometen la calidad. Por otro lado, la solucion 4, puede llegar a complicar mas el problema, ya que sumarle el paso de juntar los archivos puede derivar en perdida de pixeles, lo que no seria bueno.

Por otro lado, la solucion 1 es la mas viable y aunque sea más costosa computacionalmente, cumple todos los estándares académicos, técnicos y funcionales del proyecto.

III. Seleccion de la propuesta final

Despues de un analisis exhaustivo para la seleccion de la solucion que mejor se adapta al proyecto, se decidio usar la primera solucion, es decir la expansion de 2x2 a 4x4. Esto por diversos factores que como se menciono en el punto anterior, inclinaban mas la balanza a esa selccion como por ejemplo, el simple hecho de que a como se propuso esta solucion, es el como viene del enunciado del proyecto, lo que daria mas viabilidad y efectividad porque asi es como lo planteo el profesor, ademas de que es la solucion que da una mejor calidad de imagen y que es mas precisa. De misma manera como se indico en esta, el codigo planea hacerse de una forma simple y no tan complicada, por lo que se mantendria el modelo original de un simple archivo .img que se volveria para ensamblador un buffer de tamaño 97x97, mientras que la imagen de salida tambien en formato .img tambien va a ser una buffer de salida, pero de tamaño 289x289. La manera mas simple de implementar esto es de primeras abrir el buffer de entrada y recorrerlo copiando los valores originales en el buffer de salida para poder asi comenzar con la interpolacion, empezando a recorrer con los indices de la matriz filas y columnas, permitiendo asi saber que cada dos pares van a ir dos valores interpolados, por lo que para ese manejo se avanzan 3 posiciones para poder iniciar la siguiente matriz con los valores de la ultima columna de ese rango.

References

[1] Cloudinary, “What is Bilinear Interpolation?,” *Cloudinary Glossary*. [Online]. Available: https://cloudinary.com/glossary/bilinear-interpolation.