**Trabajo Final Arquitectura de Computadores y Laboratorio**

*Juan Martín García Patiño, Douglas Lopez Mena*

***Introducción***

El planteamiento y desarrollo de algoritmos es un trabajo que conlleva a un análisis ardúo para determinar una función específica de manera eficiente, refiriéndose a eficiencia como el tiempo que tarda en realizarse dicha función. En muchas ocasiones, la complejidad algorítmica no es suficiente para definir el desempeño total de un algoritmo o programa en general, ya que se estaría omitiendo factores vinculados a la arquitectura de hardware, como la localidad espacial, la jerarquía de memoria de nuestros equipos, las cuales influyen en el desempeño de nuestro programa.

En el presente trabajo nuestra primera base de estudio es el comportamiento y desempeño de 5 algoritmos cuya finalidad o función es invertir los colores de las imágenes, a través de el diseño de experimentos, DoE.

***Objetivos***

El trabajo final tiene como objetivo la aplicación del diseño y análisis de experimentos, se realizó con el fin de evaluar el desempeño y eficiencia de invertir el color de una matriz que representa una imagen de mapa de bits con diferentes algoritmos.

***Metodología***

Para poder tomar los datos del proyecto se tenía que buscar diferentes imágenes para luego ajustarlas a los tamaños (64 x 64, 160 x 160, 512 x 512, 1500 x 1500). Una vez ajustadas dichas imágenes, fue necesario cambiar la profundidad de bits por pixel, para esto se utilizó la herramienta Adobe Photoshop (Ps). Después de haber ajustado la profundidad de bit por píxel de cada imagen (16, 24, 32). Se utilizaron 3 tipos diferentes de imágenes, para cada profundidad. Ahora bien, siguiendo en el proceso, se nos entregó un archivo fuente con 5 versiones de algoritmos diferentes, pero que su objetivo era el mismo (invertir los colores de una imagen), escritos en el lenguaje *Matrix Laboratory* (MATLAB) con el objetivo de traducirlos al lenguaje C# y trabajar en la plataforma *Visual Studio 2019*. Antes de comenzar a tomar datos se hizo un Diseño de Experimentos utilizando la aplicación *Minitab Statistical Studio* para seguir un orden aleatorio para obtener los datos. Finalmente, se corre el programa de C# por consola, se toman los datos de tiempo que tarda en ejecución cada versión del algoritmo según el Diseño de Experimentos, esto para agilizar el proceso y al final se apilan los datos para poder procesarlos por *Minitab*.

***Análisis Preliminar***

Dado que un experimento conlleva a factores de estudio primarios, secundarios, tratamientos y todo esto para generar datos en una variable de respuesta específica, entonces los componentes del experimento son:

Factores primarios:

Estos factores se consideran las variables que son controladas por aquel que esté realizando el experimento, de las cuales me generan una variable de respuesta.

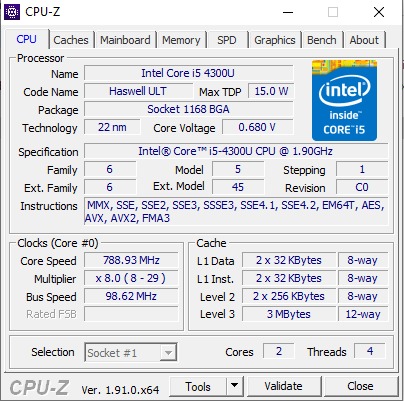
*tabla 1*

|  |  |
| --- | --- |
| Factores | Niveles |
| Tipo de algoritmo | 1. Versión 1 2. Versión 2 3. Versión 3 4. Versión 4 5. Versión 5 |
| Tamaño de la Imagen, RGB | 1. 64 x 64 2. 160 x 160 3. 512 x 512 4. 1500 x 1500 |
| Profundidad de la imagen | 1. 16 bits 2. 24 bits 3. 32 bits |

Factores Secundarios:

Son las variables que no son estudiadas en el experimento pero influyen en la variable de respuesta, sin embargo, estos factores pueden ser controlados para que el margen de error no sea un problema, es decir pueden ser eliminados o dejarlos constantes durante el experimento. Algunos de los factores secundarios del experimento fueron:

* Procesador: durante el proceso de experimentación y toma de datos, se usó un computador con un CPU Intel Core i5 de 4ta Generación. En la siguiente *gráfica 1* muestra los detalles de la jerarquía de memoria basándose en el programa CPU-Z.

*gráfica 1*

*tabla 2. Características de la jerarquía de cache de Intel Core i5*

|  |  |  |  |
| --- | --- | --- | --- |
| Cache type | Tiempo de acceso | Tamaño Cache | Tamaño de Bloques |
| L1 Data | 4 ns | 32 KB | 64B |
| L1 Inst. | 4 ns | 32 KB | 64B |
| L2 | 11 ns | 256 KB | 64B |
| L3 | 30-40 ns | 3 MB | 64B |

* Sistema operativo: Windows 10 x64
* Memoria RAM: se operó con una memoria RAM de 8GB, con un tiempo promedio de acceso de 100 ns.
* Interrupciones: una interrupción hace que el procesador detenga la tarea que estaba llevando a cabo para atender dicha interrupción, generando así un aumento en el tiempo en la variable de respuesta del experimento; para mitigar este problema, durante la ejecución del experimento no se ejecutaron tareas adicionales, simplemente se dejaron constantes.
* Programas o procesos abiertos en el computador: este factor secundario se controló de manera constante, es decir el experimento se ejecutó solamente por consola y se trató de tener el computador conectado a la corriente y con el excel abierto para el registro de los datos.
* Frecuencia del reloj: este factor mide el número de ciclos que el procesador ejecuta por segundo, medidos generalmente en Ghz (gigahercios). Esto quiere decir que es la velocidad con que nuestro computador lee las instrucciones, este factor influye en la arquitectura de hardware y por tanto en el experimento ya que es probable que si la velocidad es mayor el algoritmo termina su tarea más rápido, pero es un factor que no se estudiará, y es constante, en este caso de 1.9 Ghz.

Variable de respuesta:

La variable de respuesta para este experimento fue el tiempo de ejecución del algoritmo para invertir la imagen, dado ciertos factores.

***Diseño del Experimento***

El diseño de experimento utilizado fue el diseño factorial completo, el cual tiene como función realizar todas las posibles combinaciones de los 3 factores (algoritmo, tamaño y profundidad de imagen). Este diseño fue el más apropiado, debido a que se necesita estudiar el impacto de cada factor en conjunto o individual en la variable de respuesta. En consecuencia, es un proceso que agiliza la obtención de datos considerando todos los posibles tratamientos.

En relación con lo anterior, el experimento consta de 3 factores con los siguientes niveles para cada factor:

* Algoritmos con 5 niveles (version 1, 2, 3, 4 y 5)
* Tamaño de imagen, RGB, con 4 niveles (64 x 64, 160 x 160, 512 x 512, 1500 x 1500)
* Profundidad de imagen, RGB, con 3 niveles (16, 24, 32)

De este modo, el experimento contiene un total de 5 x 4 x 3 = 60 combinaciones.

Considerando que 60 observaciones es una muestra muy poco significativa, se decidió optar por aplicar 3 réplicas por cada tratamiento. Por lo tanto, se obtuvieron 60x3 = 180 observaciones, esto con el fin de generar mayor cantidad de datos, y así, hacer un análisis y planteamiento de hipótesis de manera más acertada y significativa.

El experimento se realizó en un mismo equipo de cómputo, teniendo así variables o factores que no son de estudios constantes, esto para disminuir el margen de error del experimento.

***Resultados***

Al obtener los datos, los resultados fueron arrojados como tiempo de ejecución total de ejecución del algoritmo dado en nanosegundos, luego se realizó una normalización para obtener el tiempo de ejecución iteración de cada ciclo teniendo en cuenta el número de accesos del algoritmo. La normalización fue calculada como el tiempo total transcurrido por el algoritmo (en nanosegundos) dividido el tiempo correspondiente al de la versión 1, es decir, para el tiempo del algoritmo 2 con tamaño 64 x 64 y profundidad 16 se divide por el tiempo de la versión 1 con los mismos factores. Esto se hizo con el fin de realizar una comparación significativa con respecto a la versión 1, ya que al tratar de hacer la comparación por otros factores no se veía diferencia, la normalización de datos no fue el más óptimo. Debido a esto, la normalización para el algoritmo 1 sus datos siempre son 1, esto quiere decir que es una recta horizontal que corta en 1

Un resúmen de los datos obtenidos se ilustra en la *tabla 1*, lo cual representa el tiempo promedio que demoró cada versión de algoritmo en ejecutar las diferentes imágenes con los distintos factores (tamaño y profundidad), cabe aclarar que los tiempos son los normalizados.

*tabla 1*

|  |  |
| --- | --- |
| General | |
| Algoritmo (Versión) | Media (ns) |
| 1 | 1,0 |
| 2 | 2,94 |
| 3 | 0,98 |
| 4 | 1,92 |
| 5 | 0,93 |

Ahora bien, para poder observar cómo en realidad los diferentes factores influyeron en la medición del tiempo en los diferentes algoritmos, la *tabla 2* representa un promedio del rendimiento de los algoritmos con respecto al tamaño de las imágenes.

*tabla 2*

|  |  |  |  |
| --- | --- | --- | --- |
| Algoritmo respecto al tamaño | | | |
| Algoritmo (Versión) | Tamaño | Media (ns) | Media total (ns) |
| 1 | 64 | 1,00 | 1,00 |
| 1 | 160 | 1,00 |
| 1 | 512 | 1,00 |
| 1 | 1500 | 1,00 |
| 2 | 64 | 3,04 | 2,91 |
| 2 | 160 | 2,82 |
| 2 | 512 | 2,90 |
| 2 | 1500 | 2,99 |
| 3 | 64 | 1,00 | 0,98 |
| 3 | 160 | 0,97 |
| 3 | 512 | 0,95 |
| 3 | 1500 | 0,99 |
| 4 | 64 | 1,94 | 1,92 |
| 4 | 160 | 1,85 |
| 4 | 512 | 1,87 |
| 4 | 1500 | 2,02 |
| 5 | 64 | 0,93 | 0,93 |
| 5 | 160 | 0,91 |
| 5 | 512 | 0,92 |
| 5 | 1500 | 0,96 |

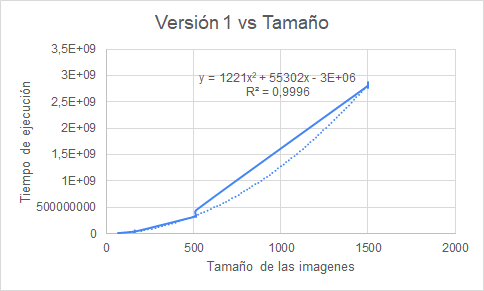
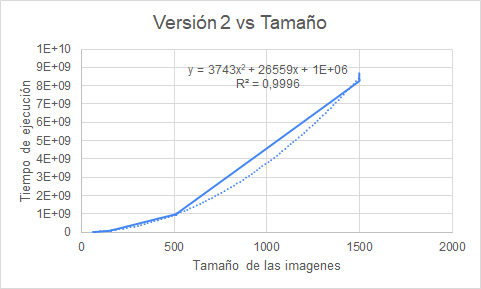
No obstante, no solo los algoritmos se vieron impactados por el tamaño de las imágenes, sino también con la profundidad de color de las mismas. Es por esto que en la *tabla 3*, se ve reflejado los resultados en base a las medias del tiempo de ejecución que emplearon los algoritmos con las diferentes profundidades de imagen.

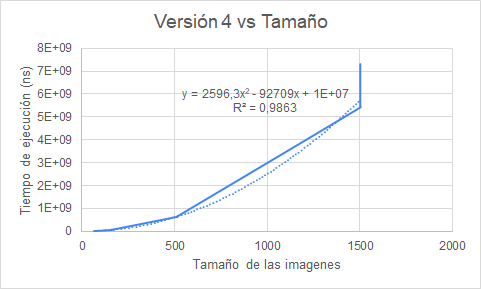
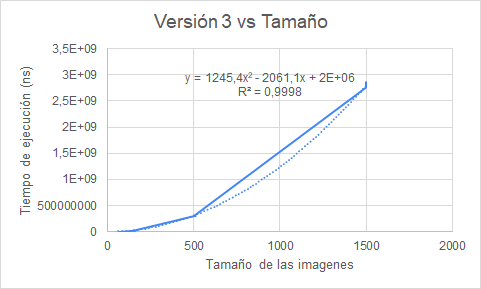
*tabla 3*

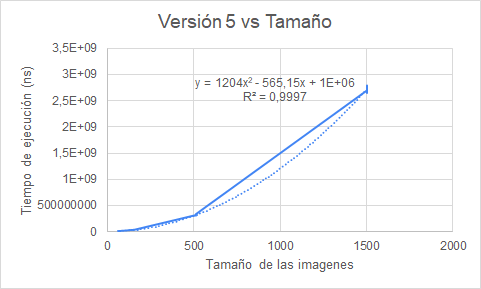
|  |  |  |  |
| --- | --- | --- | --- |
| Algoritmo/ Profundidad | 16 | 24 | 32 |
| 1 | 1,0 | 1,0 | 1,0 |
| 2 | 2,97 | 2,89 | 2,96 |
| 3 | 0,98 | 0,96 | 0,995 |
| 4 | 1,90 | 1,88 | 1,97 |
| 5 | 0,94 | 0,92 | 0,932 |

***Análisis de los resultados***

Para determinar la eficiencia de un algoritmo frente a un factor es recomendable en muchas ocasiones obtener una gráfica para analizar su rendimiento de una manera general, es por esto en las gráficas 2, 3, 4, 5, 6 se refleja el tiempo de ejecución de cada algoritmo frente a los diferentes tamaños de cada imagen.

*gráfica 2 gráfica 3*

*gráfica 4 gráfica 5*

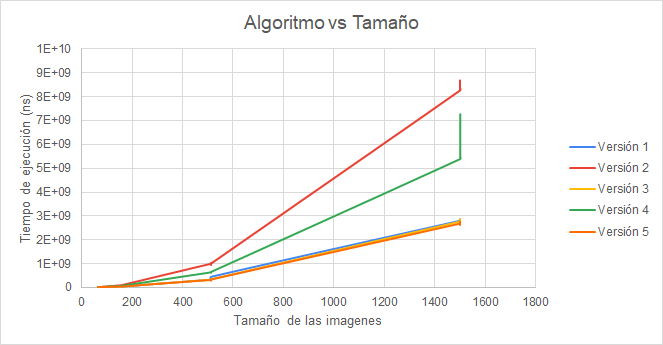
*gráfica 6*

De esta manera, cada gráfica tiene su respectiva función o línea de tendencia, tal que la variable X es el tamaño de la imagen, entonces así podemos determinar un promedio del tiempo de ejecución que tardaría cada algoritmo con un tamaño de imagen dado. Sin embargo, esta regresión polinomial de grado 2, surge debido a que cada algoritmo tiene una complejidad algorítmica de (n^2), en otras palabras, constan de dos for anidados. El valor constante que acompaña la variable de mayor grado, en este caso x^2, es el tiempo que demora el for interno en cada algoritmo, así que se puede observar que en un principio el algoritmo que toma más tiempo en ejecutar su *for* interno es el algoritmo versión 2 (gráfica 3).

No obstante, la línea de tendencia consta de otros dos términos, el segundo es de grado uno, dado que es el que determina el tiempo que demora en ejecutar el loop o for externo y el término independiente ( el que no se ve afectado por el tamaño de la imagen, X), es un tiempo que es constante y que afecta a la ejecución del algoritmo, esto puede ser por ejemplo el *clock* del procesador u otras factores que afecten el tiempo de ejecución.

Por otra parte, cada regresión polinomial representa un coeficiente de correlación, R, reflejado en cada una de las gráficas anteriores, esto indica que tan relacionado está el tamaño de la imagen con el tiempo de ejecución del algoritmo; como todas las correlaciones presentaron un valor mayor a cero, quiere decir que estas variables tienen una relación directa, es decir que si aumentamos el tamaño de la imagen, entonces el tiempo de ejecución también aumenta.

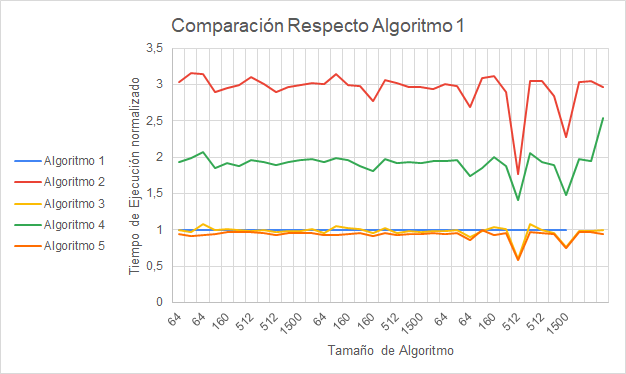
Ver las gráficas por separado dificulta un poco la comparación entre los algoritmos en base a su tiempo de ejecución, es por esto que en la gráfica 7, los algoritmos se encuentran representados en una misma gráfica.

*gráfica 7*

Ahora bien, se puede observar mejor la diferencia en tiempo de ejecución entre los algoritmos, pero cabe recalcar que aún no se puede asegurar qué algoritmo es más eficiente, dado que no se están teniendo en cuenta todos los factores primarios de estudio. Sin embargo, nos da la idea de qué versión algorítmica representa un mayor costo en tiempo de ejecución.

Dicho lo anterior, los algoritmos de versión 2 y 4, son los que tomaron más tiempo en invertir los tamaños de las diferentes imágenes (recordemos que estas gráficas sólo representan el factor algorítmico y el factor tamaño). Mientras que, las versiones 1, 3 y 5 tomaron tiempos muy similares.

Como se había mencionado anteriormente, por temas de normalización, se realizó una comparación de los algoritmos con respecto a la versión del algoritmo 1 en tiempo de ejecución, por lo que en el gráfico 8 se ilustra dicha comparación:



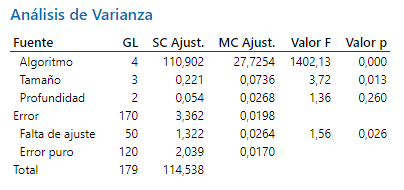
*gráfica 8. Comparación de algoritmos respecto al algoritmo 1.*

Se puede analizar que, los algoritmos 5 y 3 tienen un tiempo de ejecución muy similar con respecto al tiempo de ejecución de la versión 1. En consecuencia, para poder determinar en realidad cómo impactaron los 3 factores en los diferentes algoritmos para la variabilidad del tiempo de ejecución respecto al tiempo total de ejecución del algoritmo 1, se procede a realizar un análisis de varianza de 3 factores.

**Análisis de Varianza (ANOVA)**

Para analizar si los factores de estudio afectan a la variabilidad de la respuesta con respecto al algoritmo 1, se usó un análisis ANOVA de 3 factores, obteniendo los siguientes resultados:

*tabla 4. Modelo lineal general*

**

Teniendo en cuenta que se estudia si cada factor (algoritmo, tamaño y profundidad) influye en nuestra variable de respuesta y por tanto en la media de los datos obtenidos, se plantea las siguientes dos hipótesis:

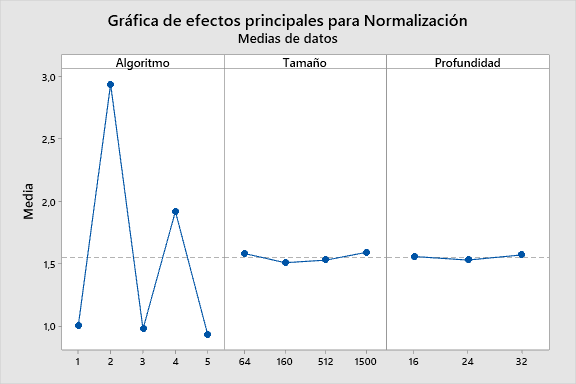
**Hipótesis nula : el factor NO influye en la media del tiempo de ejecución**

**Hipótesis alterna : el factor SÍ influye en la media del tiempo de ejecución**

Ahora bien, este análisis se trabajó con un nivel de confianza del 95%, lo que significa que si el valor P obtenido es menor al nivel de significancia del 5%, se rechaza la hipótesis nula y por tanto el factor si influiría en la variable de respuesta (tiempo de ejecución). Por consiguiente, con el análisis ANOVA se puede deducir que:

* Los factores de algoritmo y tamaño son los que influyen en la variabilidad del tiempo de ejecución en comparación con los de ejecución del algoritmo 1, es decir que su valor P es menor al 5%, por ende, estos factores de manera independientemente afecta a la media de los datos y marca su diferencia frente a los tiempos de ejecución que tiene el algoritmo 1. Por lo que si variamos cada uno de estos factores, encontramos que tiene una diferencia significativa en la variable de respuesta.
* Por otro lado, el factor de profundidad superó a más del 5%, pero no de manera exuberante, es decir que no influye en la media del tiempo de ejecución de manera significativa, por tanto es un factor que si se varía no tendría un mayor impacto en la variable de respuesta.

En consecuencia, estas deducciones se ven reflejadas en la siguiente gráfica de efectos principales de ANOVA:



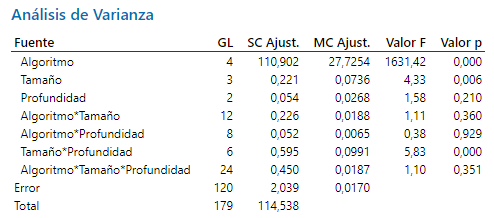
*gráfica 9. Gráfica de efectos principales*

Si se enfoca en el factor de algoritmos, se puede observar que los algoritmos 2 y 4 tienen un mayor tiempo de ejecución con respecto al tiempo del algoritmo 1, es decir que el proceso de localidad espacial que usa el algoritmo 3 y 5 son similares con respecto a 1. Dado que, el algoritmo 1 tiene un tiempo de ejecución mínimo entre los niveles del factor (*ver gráfica 7*) , entonces 2 y 5 también lo tendrían. Por esto, se puede deducir que el stride o salto de acceso a memoria para recuperar un dato de la matriz, en este caso un pixel, no es tan alto, por tanto aprovecha la localidad espacial. De ahí que, los for’s de estos algoritmos usan un recorrido por la filas en las matrices. Mientras que, los algoritmos 2 y 4 son muy distantes respecto a la media de tiempo de ejecución del algoritmo 1, debido a esto, son algoritmos que no aprovecharon su localidad espacial en su implementación, ya que para invertir un pixel su stride o acceso no es consecutivo, teniendo así un costo mayor en el tiempo de acceso, AMAT (tiempo promedio en acceso a memoria).

En adición, hubo un hecho atípico en el experimento en el que los tiempos para las imágenes de mayor tamaño tuvieron, en algunos casos, un menor tiempo de ejecución que para las imágenes de menor tamaño y se puede ver claramente en la gráfica 9: las medias de los tiempos para las imágenes de 64 x 64 y 1500 x 1500 son muy similares. Sin embargo, esto no debería ser así, porque para menor tamaño de la imagen debería necesitar menor espacio y podría ser almacenada en una caché de menor nivel y así agilizar el acceso a los datos. Por consiguiente, esto influiría con respecto a la cantidad de hits en la caché, aumentando el valor de este factor, esto por el hecho que para menor tamaño de la imagen se podría almacenar la totalidad de cada fila dentro de la caché, o por lo menos la mayor parte.

Ahora bien, las gráficas hasta ahora no considera las interacciones que pueden darse entre los diferentes factores. Es por esto, que se realizó un análisis de la varianza en los factores pero esta vez considerando las interacciones, para así poder reafirmar las deducciones previamente expuestas. Así que a continuación se presenta los resultados:

*tabla 5. Modelo lineal general de interacciones.*



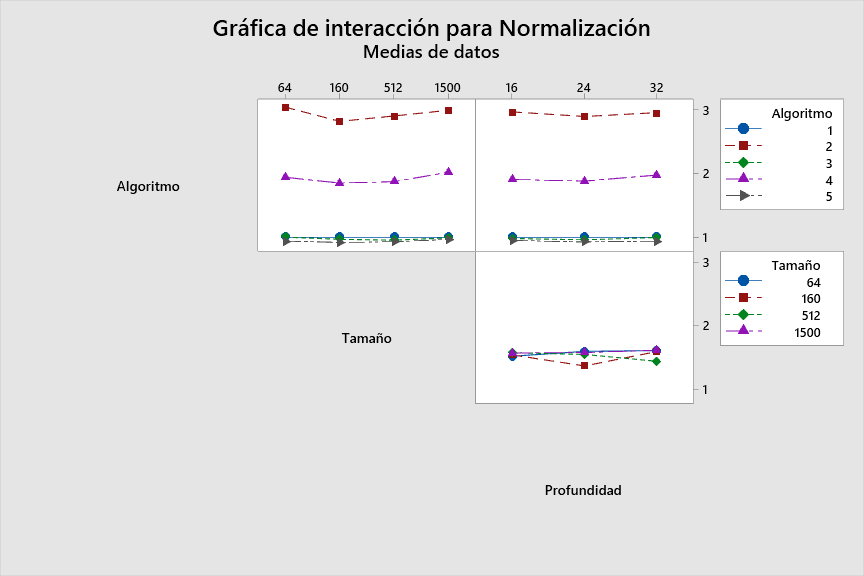
Para las interacciones ANOVA define las siguientes hipótesis:

**Hipótesis nula: No hay interacciones entre los factores**

**Hipótesis alterna: Si hay interacciones entre los factores.**

Ahora bien, considerando el 95% de significancia y el valor P de cada combinación, se puede decir que las combinaciones de factores entre Algoritmo-Tamaño, Algoritmo-Profundidad, Algoritmo-Tamaño-Profundidad, no tienen una interacción significativa, y esto sucede porque el tiempo se normalizo primordialmente en base al algoritmo 1, esto quiere decir que el tiempo de ejecución no se verá afectado por estas combinaciones, y no habrá una gran diferencia entre la variable de respuesta. En cambio, si se analiza la interacción entre Tamaño-Profundidad si afecta el resultado del tiempo de ejecución.

Estos resultados están visualmente resumidos en la siguiente gráfica:



*gráfica 10. Gráfica de interacciones para la variable de respuesta normalizada.*

Como se puede observar en la gráfica 10, las combinaciones de factores tienden a ser constantes o paralelas unas a otras, con excepción en la combinación o interacción entre tamaño y profundidad, esto implica que son combinaciones que no interactúan entre sí, por ejemplo, el algoritmo con tamaño no afecta mi variable de respuesta con los diferentes tamaños de imágenes, esto sucede ya que el tiempo de acceso a memoria es constante para cada tamaño de imagen. Sin embargo, el tiempo de ejecución medio para cada algoritmo es mucho mejor para las versiones 1, 5 y 3.

Como el uso de comparación es parte de este análisis, recordemos que se hizo con respecto a la versión 1, entonces la versión 5 es más eficiente que el algoritmo 1, dado que anteriormente se había dicho que algoritmo 1 aprovecha la localidad espacial, entonces podemos deducir que el algoritmo 5 recurre a saltos secuenciales mejores en el acceso de la memoria, esto conlleva a que el *MissRate* sea menor, por tanto, disminuye el AMAT del algoritmo.

Por otro lado, algoritmos que no aprovechan la localidad espacial como lo son las versiones 2 y 4, recurren a un mayor de costo de acceso, ocasionando una mayor cantidad de Miss al leer los píxeles de las imágenes, esto es que el algoritmo no recorre secuencialmente la memoria, entonces estos saltos de acceso aumenta el tiempo de ejecución del programa.

Una vez explicado lo anterior, sucede algo parecido con la interacción entre algoritmo y profundidad, se puede observar que, los algoritmos que no aprovechan la localidad espacial (versiones 2 y 4) continúan desaprovechando el tiempo de acceso a memoria así su profundidad esté variando, esto conlleva a que su tiempo de ejecución media, sea la misma durante la ejecución del algoritmo. En otras palabras, la profundidad o color en bits por pixel de una imagen no implica nada más que otro acceso a memoria para obtener el valor del canal, ya sea R, G o B, y estos valores igualmente son almacenados secuencialmente en la memoria, es decir que si un algoritmo no aprovecha su stride, entonces sus Hits seguirán siendo iguales, dependiendo del algoritmo y su recorrido matricial. Es por esto que la interacción es paralela, lo que significa que el algoritmo no mejora su tiempo de ejecución o su variable su respuesta con los diferentes profundidades de las imágenes. De ahí que, la comparación o la analogía que se hizo anteriormente aplica para esta interacción, ya que sus gráficas son similares, y el algoritmo 5 sigue siendo el más eficiente en invertir la matriz.

Ahora bien, es interesante analizar la interacción entre tamaño y profundidad, ya que sí existen altos y bajos en esta gráfica. De forma general, es curioso notar que para tamaños de imágenes de 160 x 160 y con profundidad 24 bits tuvo un mejor tiempo de ejecución que una de tamaño 1500 x 1500 ó 512 x 512 dada la misma profundidad; lo mismo sucede para las profundidades de 32 bits, pero en este caso la de menor tiempo fue de 512 x 512, en los otros casos el tiempo de ejecución casi el mismo. Así que, en este punto entra en participación un concepto importante de la arquitectura de hardware y es la jerarquía de memoria del equipo de cómputo.

Con relación a lo mencionado anteriormente, estas variaciones en el tiempo de ejecución ocurren debido al tamaño de nuestra memoria caché y al tamaño de nuestras imágenes. En lo que respecta a que algunos tamaños se demoran menos que otros a pesar de ser más grandes, esto es debido a que su peso en KB es cercano al de otras imágenes y son almacenados en la memoria caché nivel 1 y 2, así que cuando se accede ocurre que existan menos Miss que en una ejecución hecha previamente con un tamaño distinto. Por ejemplo, 64 x 64, 160 x 160, 512 x 512 de 16 bits de profundidad tienen un peso de 8KB, 50KB, 512KB respectivamente, por lo que las dos primeras imágenes ocupa la memoria caché nivel 1 y una 3ra parte de la memoria caché de nivel 2, esto quiere decir que la imagen de 64 x 64 va a tener más miss que una de 160 x 160 ya que la memoria caché inicialmente arrancaría en estado frío, y cuando ya ejecuta puede que la imagen de 160 x 160 ya esté en memoria, y por tanto obtenga más hits de lo esperado. Entonces una imagen de mayor de tamaño puede obtener un tiempo AMAT, o tiempo promedio de acceso a memoria menor que una de menor tamaño, teniendo en cuenta que haga buen uso de la localidad espacial.

**Conclusiones**

Para el cierre del trabajo experimental cubierto a lo largo de este documento se puede concluir que, claramente, los algoritmos 1, 3 y 5 aprovechan de manera positiva la localidad espacial, lo dicho quiere decir que el tiempo de ejecución total es menor para estas versiones mencionadas que para el resto. Esto se ve muy claro en las gráficas 8 y 9, a pesar de ser una comparación de tiempos con el algoritmo 1. En adición, el factor de profundidad no tuvo una notable importancia en el experimento, diferente a lo que se esperaba, aunque esto pudo ser debido a la manera de procesar las imágenes, ya que se usó una librería específica de C# para el manejo de imágenes como matrices de bits (Bitmap). Por otro lado, se obtuvo un grupo datos atípicos en cuestión del tamaño para imágenes de 512 x 512 y 1500 x 1500, que mostró un gran cambio en los tiempos en todos los algoritmos para la profundidad de 32 bits, este suceso es apreciable en la gráfica 8. Finalmente, el factor que más afectó la variable de respuesta fue la versión de los algoritmos, mientras que para el tamaño y la profundidad de las imágenes fueron realmente poco significante para el tiempo de ejecución total y, aunque el factor de tamaño no fue muy significante, se pudo ver que el tiempo para los tamaños 64 x 64 y 1500 x 1500 fueron similares lo que implica que el MissRate para el primer tamaño fue mayor que para el segundo.