Comparación de algoritmos para invertir el color de una imagen

Cristhian Eduardo Castillo Meneses
cristhian.castillo1@correo.icesi.edu.co
Kevin Gianmarco Zarama Luna
zaramaluna1999@hotmail.com

June 1, 2020

Abstract

El trabajo final tiene como objetivo la aplicación del diseño y análisis de experimentos con el fin de evaluar el desempeño de invertir el color de una matriz que representa una imagen de mapa de bits. Cuantificar y explicar el impacto en el desempeño que tiene 5 versiones equivalentes de algoritmos en varían en la localidad espacial al recorrer sus píxeles. También se desea estudiar el impacto del tamaño de las imágenes en el tiempo de respuesta normalizado en la ejecución de la operación de invertir el color. Otra variable de interés es la afectación de la profundidad de color en el tiempo de ejecución, para ello se utilizará imágenes de 8,16y 32 bits por canal RGB.
Keywords: *RGB; Images; Algoritmos; Principio de ocalidad.*

1 Introducción

Los algoritmos se enfocan a cumplir con una determinada función, hay diferentes algoritmos que pueden resolver un mismo problema diferenciándose en la forma en que lo hacen. Por eso nace la necesidad de comparar los algoritmos para saber cuales son los más eficientes ya sea en complejidad temporal, complejidad espacial, tiempo de ejecución.

Saber el rendimiento de los algoritmos siempre ha sido una tarea muy importante para poder seleccionar el mejor algoritmo dada una situación particular o el que mejor se adapte en la mayor cantidad de casos.

En este documento se compararán 5 algoritmos que tienen como objetivo invertir los colores de unas imágenes seleccionadas, para comparar el tiempo de ejecución de cada uno de los algoritmos; también se tiene en cuenta el tamaño y la profundidad de color de las imágenes.

2 Objetivo

El experimento tiene como objetivo la aplicación del diseño y análisis de experimentos con el fin de evaluar el desempeño de invertir el color de una matriz que representa una imagen de mapa de bits.

3 Definición del Experimento

Los componentes del experimento son:

3.1 Variable de Respuesta

La variable de respuesta definida en este experimento es el tiempo de ejecución[1].

3.2 Factores Primarios

Los factores de primarios son las variables que pueden ser controladas, de las cuales se quiere ver el impacto que tienen sobre la variable de respuesta.

Table 1: Factores Primarios

Variable	Niveles
Algoritmo	Algoritmo A
	Algoritmo B
	Algoritmo C
	Algoritmo D
	Algoritmo E
Tamaño de la Imagen	64x64
	160x160
	512x512
	1500x1500
Profundidad del píxel	8 Bits
	16 Bits
	32 Bits

3.2.1 Algoritmo

Se implementan 5 algoritmos que cuentan con una complejidad algorítmica de:

$$O(n^2)$$

A pesar de que los algoritmos tienen una misma complejidad algorítmica, algunos usan de forma distinta el Principio de Localidad[2]

[1] El tiempo de ejecución es el tiempo que tarda un programa en ejecutarse en un sistema operativo

[2] Es un fenómeno según el cual, basándonos en el pasado reciente de un programa podemos predecir con una precisión razonable qué instrucciones y datos utilizará en un futuro próximo

Algoritmo A: El algoritmo A recorre la matriz de píxeles por filas, apoyándose del principio de Localidad para aumentar el tiempo de ejecución. Al recorrer la matriz de esta forma hace que la tasa de missrate en comparación a los otros algoritmos sea menor.

```
for y in range(len_y):
    for x in range(len_x):
        data[x,y] = (255-data[x,y][0], data[x,y][1], data[x,y][2])
        data[x,y] = (data[x,y][0], 255-data[x,y][1], data[x,y][2])
        data[x,y] = (data[x,y][0], data[x,y][1], 255-data[x,y][2])
```

Figure 1: Versión A del algoritmo para invertir el color de una imagen

Algoritmo B: recorre la matriz de píxeles por filas, al igual que el algoritmo A, pero este lo recorre con 3 for, uno para cambiar cada bit, este algoritmo tiene la misma tasa de missrate que el algoritmo A, ya que recorre la matriz por filas, pero se demora mas debido a que debe recorrer la matriz 3 veces para cambiar de color cada píxel

```
for y in range(len_y):
    for x in range(len_x):
        image[x,y] = (255-image[x,y][0], image[x,y][1], image[x,y][2])

for y in range(len_y):
    for x in range(len_x):
        image[x,y] = (image[x,y][0], 255-image[x,y][1], image[x,y][2])

for y in range(len_y):
    for x in range(len_x):
        image[x,y] = (image[x,y][0], image[x,y][1], 255-image[x,y][2])
```

Figure 2: Versión B del algoritmo para invertir el color de una imagen

Algoritmo C: recorre la matriz de píxeles por columnas, esto hace que el algoritmo tenga un missrate alto, ya que tiene que recorrer la matriz de forma tal que tiene q cambiar constantemente entre las filas del arreglo, este algoritmo tiene un missrate de 1.

```python
for x in range(len_x):
    for y in range(len_y):
        image[x,y] = (255-image[x,y][0], image[x,y][1], image[x,y][2])
        image[x,y] = (image[x,y][0], 255-image[x,y][1], image[x,y][2])
        image[x,y] = (image[x,y][0], image[x,y][1], 255-image[x,y][2])
```

Figure 3: Versión C del algoritmo para invertir el color de una imagen

Algoritmo D: recorre la matriz de píxeles por filas, pero tiene un missrate mas alto, debido a que usa dos for, en el primero recorre la matriz completa pero solo cambia el rojo de los píxeles, y en otro for recorre los otros dos colores de la matriz, lo cual hace que se tenga un missrate alto, ya que tiene que recorrer la matriz dos veces y cambiar entre sus valores.

```python
for y in range(len_y):
    for x in range(len_x):
        image[x,y] = (255-image[x,y][0], image[x,y][1], image[x,y][2])

for y in range(len_y):
    for x in range(len_x):
        image[x,y] = (image[x,y][0], 255-image[x,y][1], image[x,y][2])
        image[x,y] = (image[x,y][0], image[x,y][1], 255-image[x,y][2])
```

Figure 4: Versión D del algoritmo para invertir el color de una imagen

Algoritmo E: recorre la matriz de píxeles de forma tal que lo hace en grupos de 2 x 2 píxeles, lo cual hace que el missrate de este algoritmo sea algo alto, ya que tiene que cambiar entre las filas y columnas de la matriz constantemente

```python
for y in range(0, len_y, 2):
    for x in range(0, len_x, 2):
        image[x,y] = (255-image[x,y][0], image[x,y][1], image[x,y][2])
        image[x,y] = (image[x,y][0], 255-image[x,y][1], image[x,y][2])
        image[x,y] = (image[x,y][0], image[x,y][1], 255-image[x,y][2])

        image[x+1,y] = (255-image[x+1,y][0], image[x+1,y][1], image[x+1,y][2])
        image[x+1,y] = (image[x+1,y][0], 255-image[x+1,y][1], image[x+1,y][2])
        image[x+1,y] = (image[x+1,y][0], image[x+1,y][1], 255-image[x+1,y][2])

        image[x,y+1] = (255-image[x,y+1][0], image[x,y+1][1], image[x,y+1][2])
        image[x,y+1] = (image[x,y+1][0], 255-image[x,y+1][1], image[x,y+1][2])
        image[x,y+1] = (image[x,y+1][0], image[x,y+1][1], 255-image[x,y+1][2])

        image[x+1,y+1] = (255-image[x+1,y+1][0], image[x+1,y+1][1], image[x+1,y+1][2])
        image[x+1,y+1] = (image[x+1,y+1][0], 255-image[x+1,y+1][1], image[x+1,y+1][2])
        image[x+1,y+1] = (image[x+1,y+1][0], image[x+1,y+1][1], 255-image[x+1,y+1][2])
```

Figure 5: Versión E del algoritmo para invertir el color de una imagen

3.2.2 Tamaño de Imagen

Se quiere conocer el impacto que tiene el tamaño de imagen en el tiempo de ejecución de los algoritmos.

3.2.3 Profundidad del píxel

La profundidad del píxel[3] es un aspecto que podría influir en el desempeño de invertir el color de una matriz.

3.3 Factores Secundarios

Los factores secundarios son factores que influyen en la variable de respuesta pero que no son relevantes para el estudio. Este tipo de factores pueden ser controlados para no afectar el experimento. A continuación se en listan los factores Secundarios.

Hardware y SO El procesador, la memoria RAM, la memoria cache, el sistema operativo son factores que pueden influir en en el experimento por lo que se decidió usar una unidad experimental con las siguientes características.

[3]se refiere a la cantidad de bits de información necesarios para representar el color de un píxel en una imagen digital

Table 2: Factores Primarios

Hardware/SO	Especificación
Procesador	AMD Ryzen 5 Mobile 3550H
Core Speed	2894.13 MHz
Bus Speed	99.8 MHz
Memoria Cache	L1 Data: 4x32 KB (8 ways)
	L1 Inst: 4x64 KB (4 ways)
	Nivel 2: 4x512 KB (8 ways)
	Nivel 3: 4MB (16 ways)
Cores	4

3.4 Factores no controlables

El número de procesos es un factor que puede afectar el rendimiento en el tiempo de ejecución de los algoritmos. Debido a que este factor no se puede controlar en una unidad experimental no especialidad, la medida que se tomó para mitigarla es no correr procesos como en navegador, otros programas para intentar que la menor cantidad posible de procesos puedan interrumpir con el experimento.

4 Diseño del Experimento

Se utilizó un diseño factorial completo, es decir, todas las combinaciones de los niveles de los factores primarios. Se decide tomar este diseño para ver la influencia de los factores primarios sobre la variable de respuesta. Lo cual nos genera un total de 60 tratamientos:

5 algoritmos x 4 Tamaños x 3 Profundidad bits = 60 tratamientos

Se decidió realizar 3 repeticiones y 3 replicas. Para un total de 540 datos recolectados.

5 Ejecución del experimento

Para el experimento los algoritmos fueron programados en Python y se útilizó la consola de Powershell para correr cada uno de los algoritmos.

En cada replica se cierra la ventana de Powershell y se abre una nueva para. El tiempo de ejecución se midió en segundos usando la librería time de Python.

Se cierran también todas las aplicaciones y procesos de segundo plano, no indispensables para el sistema operativo, que podrían afectar el rendimiento de los algoritmos.

6 Análisis de Resultados

Una vez obtenidos los datos, se realiza una normalización. Por lo que se divide el tiempo de ejecución por el alto y ancho de la imagen.

Con los datos una vez normalizados se crea un proyecto en Minitab para analizar los datos obtenidos.

6.1 Análisis de la Varianza (ANOVA)

Se realizó un ANOVA de 3 factores obteniendo los siguientes resultados:

Análisis de varianza de Tiempo

Fuente	GL	SC	MC	F	P
Algoritmo	4	2,0222	0,5055	0,009	0,000
Tamaño imagen	15	888,0080	59,2005	5533,480	0,000
Resolucion	40	0,4279	0,0107	0,838	0,736
Error	120	1,5326	0,0128		
Total	179	891,9908			

Figure 6: ANOVA de 3 factores (Algoritmo, Tamaño y profundidad del píxel)

Para su interpretación se tienen las siguientes hipótesis:
Hipótesis nula: el factor NO influye en la media de los datos
Hipótesis alterna: el factor SI influye en la media de los datos

Hay que recordar que en el ANOVA si el valor p es menor que el nivel de significación (5%) se rechaza la hipótesis nula. Entonces con un nivel de confianza del 95% se puede decir que:

Los factores del algoritmo y el tamaño de la imagen tienen una influencia en la media de los datos (tiempo de ejecución). Por otro lado la resolución de la imagen no afecta significativamente en la media de los datos.

7 Interpretación

Se compara como influye los factores en la variable de respuesta (tiempo de ejecución).

A continuación se muestra los gráficos que muestran la influencia en los tiempos de ejecución.

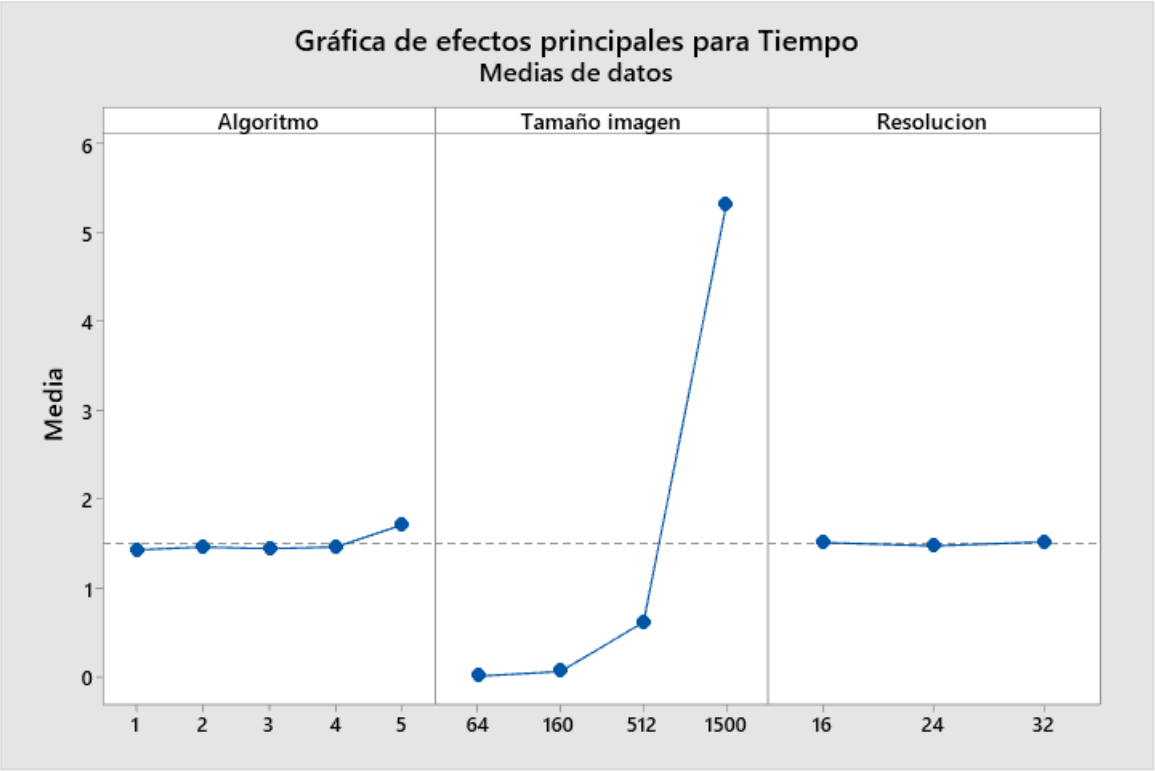

Figure 7: gráfica de efectos principales para tiempo de los datos de tiempo 1. Respuestas: Tiempo. Factores: algoritmo, tamaño imagen y resolución.

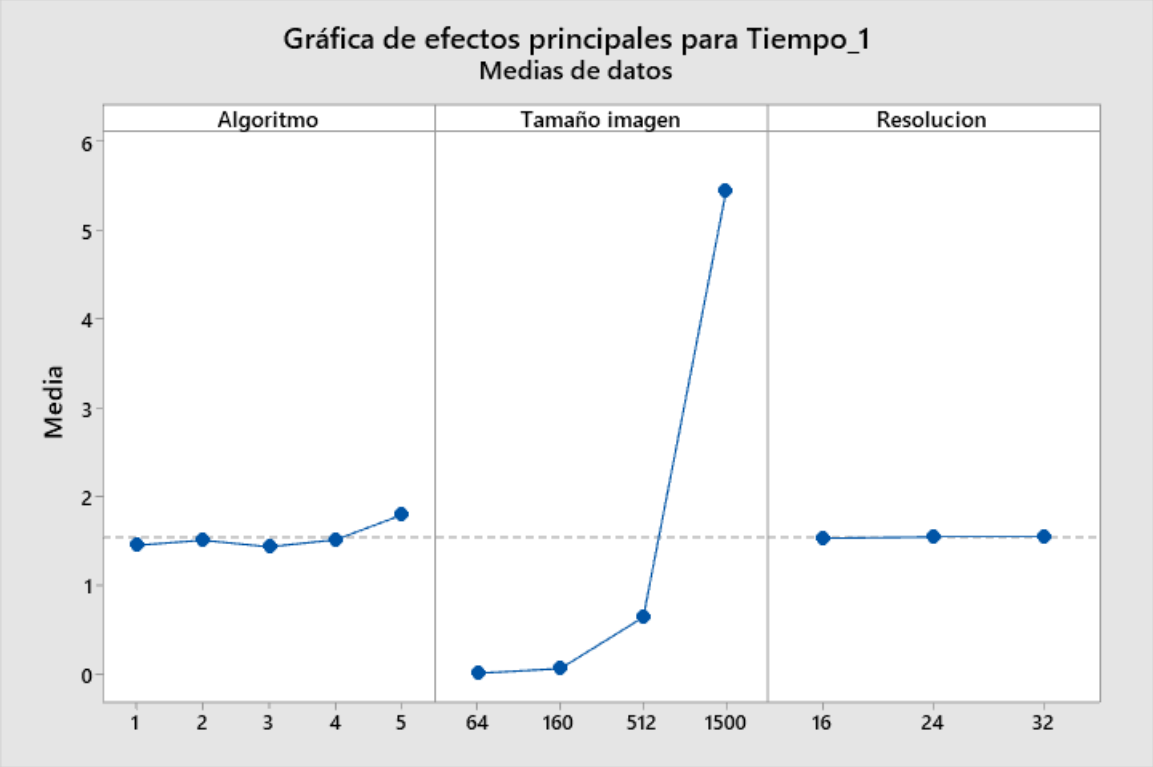

Figure 8: gráfica de efectos principales para tiempo de los datos de tiempo 2. Respuestas: Tiempo. Factores: algoritmo, tamaño imagen y resolución.

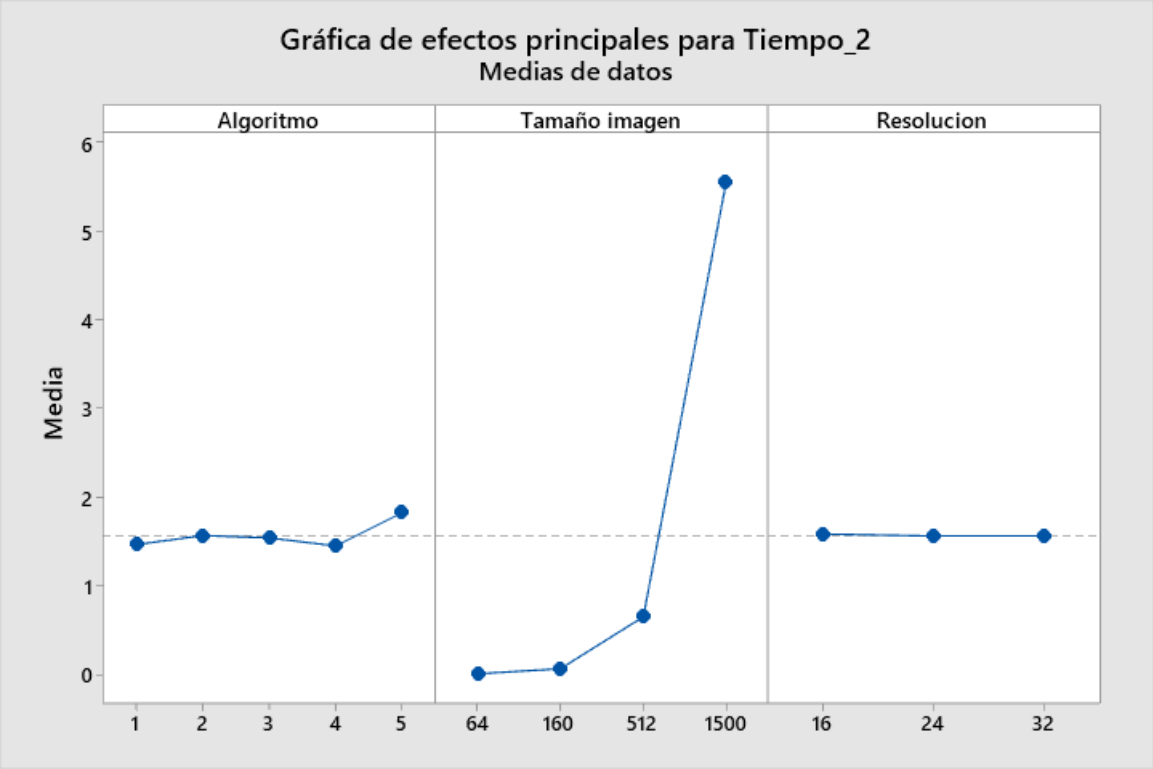

Figure 9: gráfica de efectos principales para tiempo de los datos de tiempo 3. Respuestas: Tiempo. Factores: algoritmo, tamaño imagen y resolución).

El factor que se cree más influye es el algoritmo por lo cual decidimos ver el efecto de este factor sobre la variable de respuesta:

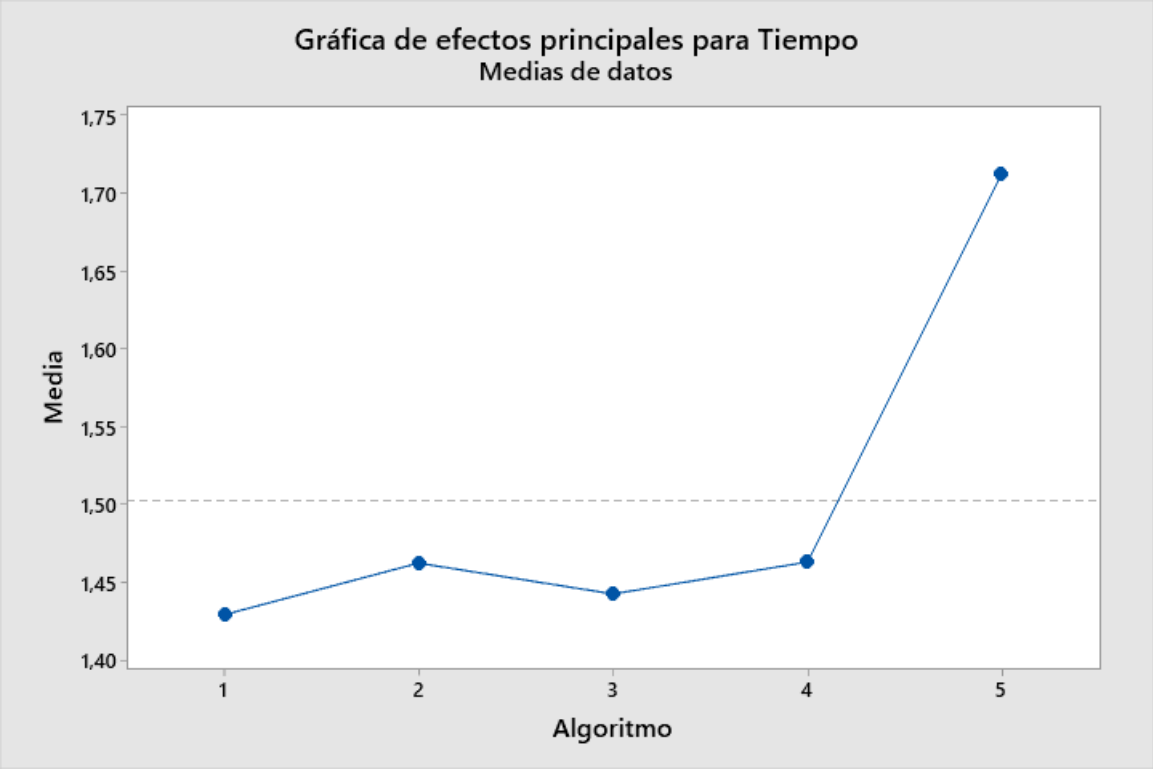

Figure 10: ampliación de la gráfica de efectos principales para el tiempo de los datos de tiempo 1. Respuesta: tiempo. Factores: algoritmo.

También es importante ver la interacción de los factores con la variable de respuesta, por lo que se gráfica también:

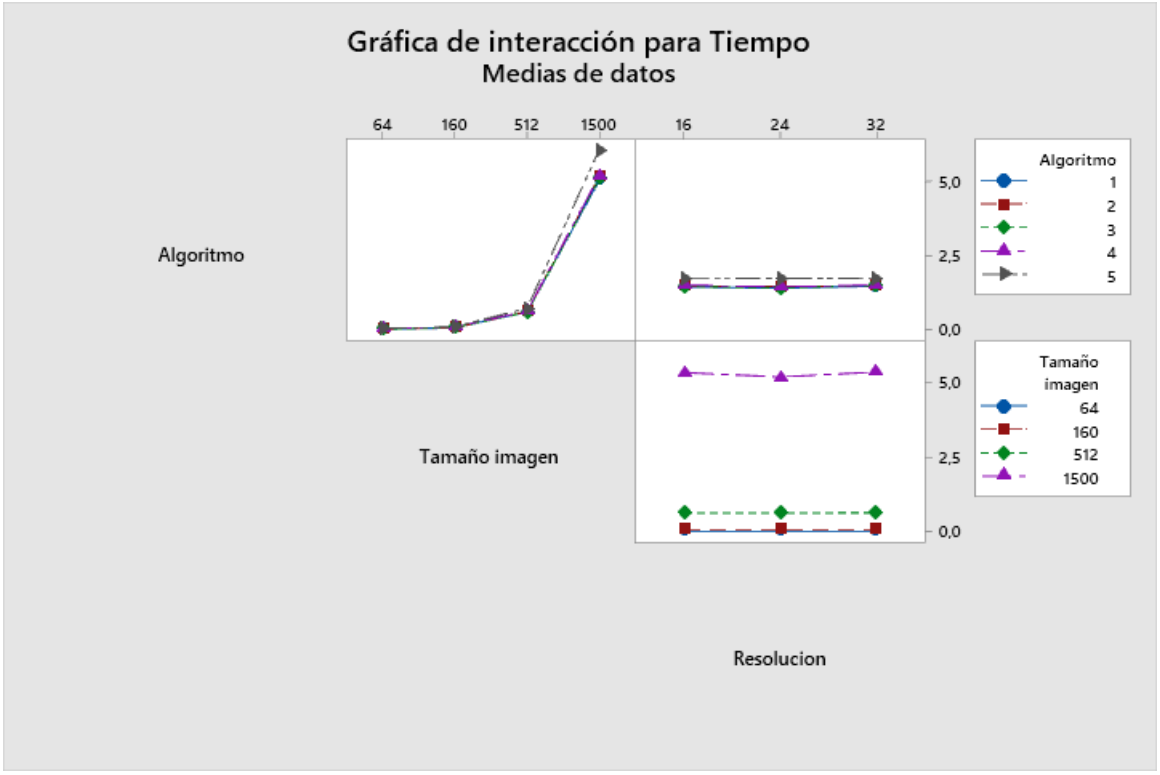

Figure 11: gráfica de interacción para tiempo de los datos de tiempo 1. Respuestas: Tiempo. Factores: algoritmo, Tamaño imagen y resolución

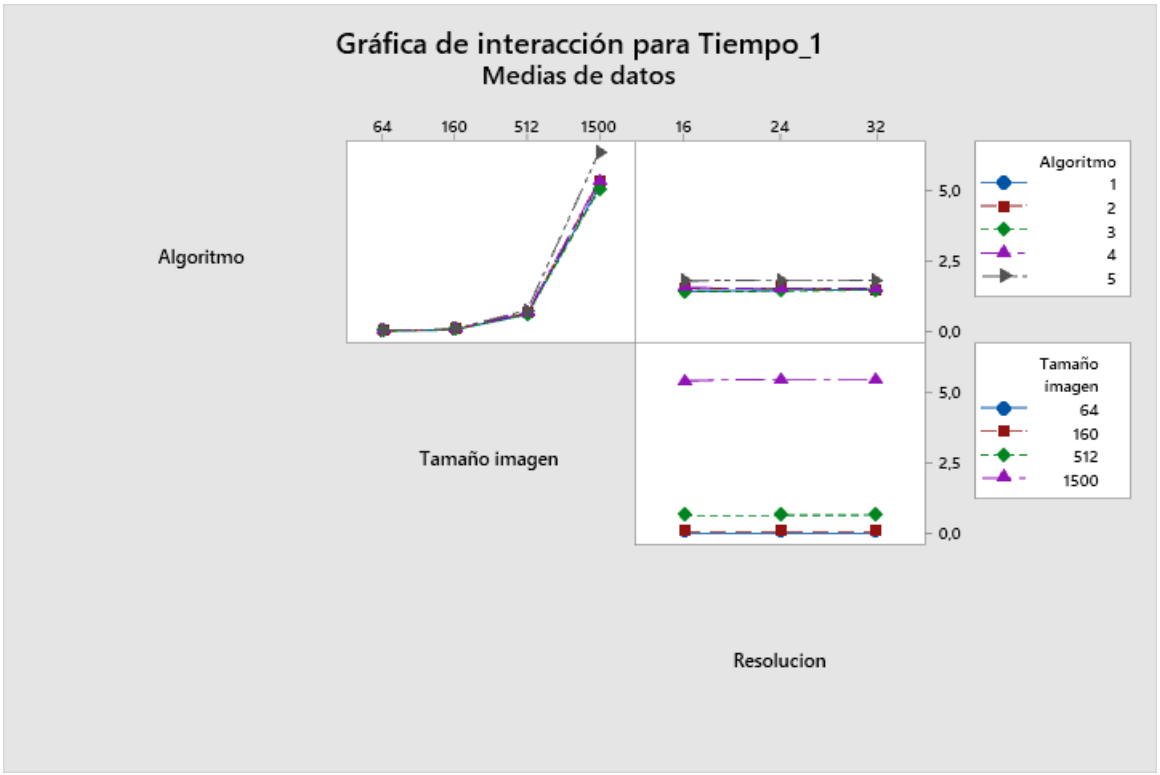

Figure 12: gráfica de interacción para tiempo de los datos de tiempo 2. Respuestas: Tiempo. Factores: algoritmo, Tamaño imagen y resolución

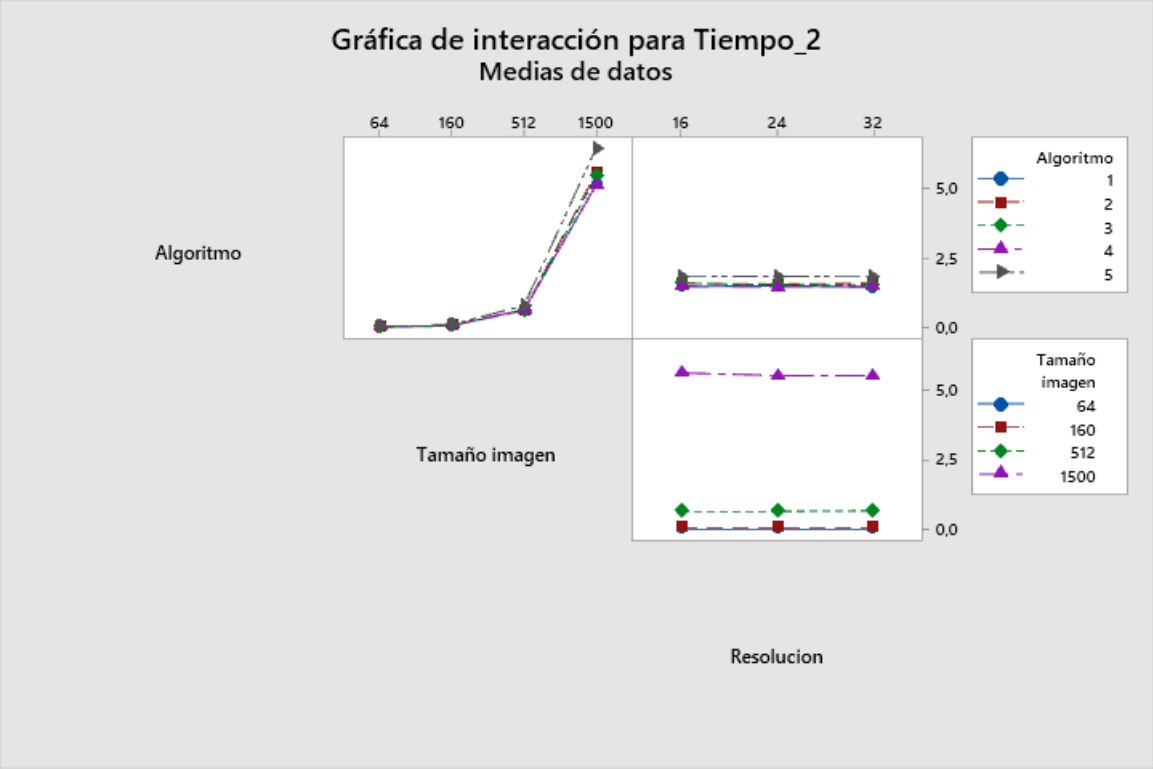

Figure 13: gráfica de interacción para tiempo de los datos de tiempo 3. Respuestas: Tiempo. Factores: algoritmo, Tamaño imagen y resolución

Con esta información se puede decir que uno de los datos que menos influye en el tiempo de ejecución es la profundidad de los píxeles de las imágenes, por eso, las gráficas que relacionan la resolución con el tiempo de ejecución son rectas; el algoritmo elegido para cambiar los colores de lo píxeles es una característica que no influye drásticamente en el tiempo de ejecución, esto se puede observar en las gráficas de interacción para tiempo (gráfica 11, gráfica 12, gráfica 13) en el hecho de que las rectas no se separan mucho entre sí.

El tamaño de la imagen fue la característica que mas influyo en los tiempos de ejecución fue el tamaño de la imagen, esto se observa en las gráficas de interacción para tiempo (gráfica 11, gráfica 12 y gráfica 13), ya que cuando el tamaño de la imagen influye en la gráfica se ve que hay un cambio considerable a medida de que el tamaño aumenta.

En relación con los algoritmos elegidos, los algoritmos 1, 2, 3, 4, no tienen muchas diferencias de tiempo entre ellas, en cambio el algoritmo 5 si tiene una diferencia considerable con respecto a los otros algoritmos, como se puede observar en la figura 10; la diferencia del tiempo de ejecución del algoritmo 5, se debe a que no aprovecha el uso de la localidad espacial, ya que este algoritmo va trabajando en bloques de 4 píxeles en un arreglo de 2x2 píxeles y esto hace que el algoritmo cambie mucho entre las filas y columnas de la matriz, lo cual hace que tenga un missrate mucho mayor en comparación a los otros algoritmos.

El mejor algoritmo es el algoritmo 1, debido a que recorre la matriz de forma tal que primero recorre las filas de la matriz, y esto hace que sea mas eficiente para recorrer la matriz que el algoritmo 3, el cual tiene la misma estructura que el algoritmo, pero accede

a los datos primero a las filas y después intercambiando entre las columnas, lo cual hace que el algoritmo tengo mas missrate que el algoritmo 1; los algoritmos 2 y 4 tienen muchas similitud de tiempo de ejecución entre sí, las diferencias de estos algoritmos son que tienen mayor localidad temporal, ya que tienen 3 y 2 for respectivamente, los cuales acceden a varias veces a los mismos píxeles, lo cual aumenta el tiempo de ejecución.

8 Conclusiones

El principio de localidad espacial es muy importante para hacer que los algoritmos sean más eficiente, debido a que aunque los algoritmos puedan tener la misma complejidad algorítmica el principio de localidad puede influenciar en gran medida en el tiempo de ejecución de los algoritmos, que es vital para cuando queremos correr un algoritmo en un conjunto muy grandes de datos sin tardar meses o años.

References

- *get image size (width, height) with python, opencv, pillow (pil).* (n.d.). Retrieved from `https://note.nkmk.me/en/python-opencv-pillow-image-size/`
- *how to get an array from rgb values of a bitmap image?* (n.d.). Retrieved from `https://stackoverflow.com/questions/21885713/how-to-get-an-array-from-rgb-values-of-a-bitmap-image`

William., S. (2010). *Computer organization and architecture : designing for performance*. Prentice Hall.