Compresión de Colores utilizando K-means

Rodrigo Andrés Muñoz Gomez
Pontificia Universidad Católica de Valparaíso Chile
rodrigo.munoz.g@mail.pucv.cl

***Abstract*—La compresión de colores es un método ampliamente utilizado tanto en áreas centradas en el procesamiento de imágenes y señales, como en el funcionamiento cotidiano de dispositivos electrónicos. Este artículo se enfoca en el uso del algoritmo k-means con el propósito de comprender y demostrar en la práctica su funcionamiento en la compresión de colores y, por ende, en la reducción del consumo de espacio en memoria. Además, se emplea un algoritmo complementario para la descompresión.**

***Index Terms*—Compresión, Descompresión, K-means, Colores**

I. Introducción

Partiendo de una imagen de entrada, el algoritmo K-means tiene la responsabilidad de realizar la compresión de colores a través de un parámetro preestablecido, denotado como "K". Utilizando este parámetro, se buscan K colores mutuamente excluyentes mediante un proceso iterativo. Durante este procedimiento, se calculan las distancias entre cada color de la imagen original y los K grupos definidos previamente. A partir de estas distancias, se asignan los colores de la imagen a los clusters representativos correspondientes. Finalmente, se calcula el valor promedio de todos los puntos pertenecientes a un cluster para reajustar el propio cluster, dando así inicio a la siguiente iteración. [1]

Algorithm 1 Algoritmo K-means General

Require: X: Conjunto de colores, K: Número de clusters
1: Inicializar centroides $\{\mu_1, \mu_2, \ldots, \mu_K\}$ aleatoriamente
2: **repeat**
3: **for** cada color $x_i \in X$ **do**
4: Calcular $j = \arg\min_k \|x_i - \mu_k\|^2$
5: Asignar x_i al cluster j
6: **end for**
7: **for** cada cluster C_k **do**
8: Actualizar centroide $\mu_k = \frac{1}{|C_k|} \sum_{x_i \in C_k} x_i$
9: **end for**
10: **until** no hay cambios en la asignación de colores

En general, la compresión de colores se emplea ampliamente en diversas áreas, como el procesamiento digital, la compresión y la informática gráfica, entre otros campos. Por esta razón, en el presente artículo se desarrollarán y examinarán dos algoritmos. Uno de ellos tiene la capacidad de comprimir colores de una imagen con dimensiones $M \times N \times 3$ a una matriz $M \times N$, la cual almacena los índices de los K colores otorgados por K-means. Estos colores también serán devueltos en formato matricial, en el cual cada fila representa una tripleta RGB. Por otro lado, se presenta un algoritmo con la capacidad de llevar a cabo la descompresión de esta información comprimida.

II. Manejo de datos e Inicializacion de Centroides

El algoritmo K-means depende directamente de la cantidad de datos con los que opera. En otras palabras, una mayor cantidad de datos implicará un mayor cómputo y tiempo invertido en la ejecución del programa. Por lo tanto, a partir de [2], podemos recurrir al muestreo de datos. Las imágenes digitales podrían llegar a tener una cantidad de colores equivalente a la cantidad de píxeles si estos son menores a 2^{24}. Sin embargo, para la mayoría de las imágenes, el número de colores que poseen es mucho menor que la cantidad de píxeles en la misma imagen. Por ende, se concluye que en la imagen existen colores repetidos.

(a) Journey (600 X 600) (b) The Outfield (600 X 600)

(c) Lena (512 X 512) (d) Mandrill (512 X 512)

Fig. 1: Ground Truth Images.

Para abordar esto, es necesario emplear únicamente los colores únicos de una imagen, agilizando así el proceso de cálculo. La figura 1 muestra las cuatro imágenes originales utilizadas como pruebas. [2]

Enfocándonos en una de ellas, como por ejemplo **(a) Journey**, podemos observar que esta imagen contiene un total de 360000 colores, pero solo 158446 de ellos son únicos. Esto

significa que solamente el 44% de los colores presentes en la imagen son únicos. Sin embargo, uno de los inconvenientes de este enfoque es que se pierde el peso de cada color (ya que los colores que se repiten más tienen un impacto estadístico mayor). Para abordar este problema, se propone asignar un valor a cada color de manera que se evite introducir errores en la imagen comprimida.

Otro de los problemas a los que se enfrenta el algoritmo k-means es su dependencia y sensibilidad a la inicialización. Por ello, se implementa el algoritmo inicializador de centroides k-means++, que selecciona el primer centroide de manera aleatoria a partir de los colores de la imagen, y los restantes se obtienen a través de la probabilidad $P(x) = \frac{D(x)}{\sum_{x' \in X} D(x')}$, donde $D(x)$ denota la mínima distancia entre un punto x en la imagen y los centroides seleccionados previamente.

Aunque el algoritmo k-means++ inicializa de manera eficiente los centroides, agrega un tiempo y cómputo significativos al algoritmo general. Por esta razón, se decide utilizar una fracción equivalente al número de centroides deseados multiplicada por un factor de 6. Esto agilizará el algoritmo general a cambio de reducir el cómputo y, en algunos casos, aumentar el error. [2]

III. Compresión y Descompresión de Colores

Algorithm 2 Compresión de Colores

Input: $Imagen\ Original\ m \times n \times 3, K$
Output: $Imagen\ Compresa\ m \times n,\ Colores\ K \times 3$

Imagen Vectorizada[$ImVec$] ← Imagen
Colores Unicos[cu], **Peso Colores**[pc] ← ImVec;
Colores Ajustados [ca] ← $cu(:,:) \times pc(:)$
K Centroides a Partir de K-means ++[c_k] ← CU;
for $j = 1{:}10$ **do**
    $Dist[1:K]$ = Distancias(cu,$c_{\in 1,2,..,k}$)
    Índices = $MinimaColumnaPorFila(Dist)$;
    for $n = 1:K$ **do**
        $Indices_{c_k}$ = find(Indices == c_k)
        a = Sum($ca(Indices_{c_k}, :)$);
        b = Sum($pc(Indices_{c_k}, :)$);
        c_k = a/b;
    end for
end for
Imagen compresa = $reshape(Indices, m, n)$
Colores = $c(1:k, :)$

El algoritmo Descrito toma como entrada una imagen original de dimensiones m x n x 3 (que representa los píxeles RGB), y un valor K que determina la cantidad de colores a usar en la imagen comprimida. Primero, el algoritmo vectoriza la imagen original y encuentra los colores únicos presentes en ella, junto con sus respectivos pesos (repeticiones por color). Luego, ajusta los colores únicos multiplicando sus valores por los pesos correspondientes. Se seleccionan K centroides a partir del algoritmo k-means ++, en donde el primer centroide se obtiene aleatoriamente desde colores únicos, luego para los centroides restantes se seleccionan a partir de la probabilidad generada a partir del calculo de la distancia entre cada punto y su centroide mas cercano previamente creado. Posteriormente, se inicia el bucle for con 10 iteraciones mediante el cual se obtienen las distancias mínimas entre cada punto y cada centroide de manera matricial para así extraer el índice del centroide con la distancia mínima para cada punto. Dado esto, comienza un bucle que determina la media para todos los puntos asociados en cada centroide para así reorganizarlos y entrar en la siguiente iteración. Finalmente, se reorganiza el vector a modo de obtener una matriz m x n con cada índice referente a un elemento del vector que contiene los centroides.

Algorithm 3 Descompresión

Input: $Imagen\ Compresa\ m \times n,\ Colores\ K \times 3$
Output: $Imagen\ Reconstruida\ m \times n \times 3$

Imagen Vectorizada[$ImVec$] ← Imagen Compresa
Imagen Reconstruida[ImRec] = $Colores(ImVec, :)$;
Imagen Reconstruida = $reshape(ImRec, m, n, 3)$;

Este algoritmo realiza la descompresión de una imagen previamente comprimida utilizando el proceso explicado en el algoritmo Descompresión. Toma como entrada una imagen comprimida de dimensiones m x n, que contiene índices de colores asignados a cada píxel, y una paleta de colores de dimensiones K x 3. En primer lugar, el algoritmo toma la imagen comprimida y la vectoriza en un arreglo unidimensional. Luego, utilizando esta vectorización, recupera los valores de los colores originales de la paleta de colores en función de los índices almacenados en la imagen comprimida. Estos valores recuperados se utilizan para construir una imagen reconstruida en forma de matriz m x n x 3 .

IV. Metricas Utilizadas

A. Mean Square Error

El MSE mide la diferencia promedio al cuadrado entre los valores reales y los valores luego de de comprimir la imagen. Un valor de MSE más bajo indica una descompresión de mayor calidad, ya que refleja una similitud más cercana entre las imágenes.

$$MSE = \frac{1}{CMN} \sum_{c=1}^{C} \sum_{m=1}^{M} \sum_{n=1}^{N} (C_c(m,n) - C_c^{'}(m,n))^2 \quad (1)$$

donde C es el número de canales de color, M y N son las dimensiones de la imagen, y $C_c(m,n)$ y $C_c^{'}(m,n)$ representan los valores de píxeles correspondientes en los canales c de ambas imágenes. [3]

B. Structural Similarity Index

El SSIM (*Structural Similarity Index*) compara la similitud estructural de las imágenes, con valores cercanos a 1 indicando una mayor similitud. Se calcula de la siguiente manera:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad (2)$$

SIMM Emplea medidas estadísticas como promedios (brillo promedio de píxeles), varianzas (variación en intensidades) y

covarianzas (cómo las intensidades varían juntas) para calcular la similitud perceptual. Esto permite evaluar cómo los cambios en estas propiedades influyen en la similitud visual entre las imágenes. [3]

V. Resultados Experimentales y Discusiones

La tabla I muestra los resultados de MSE, SSIM, Porcentaje de compresión de tamaño, Porcentaje de compresión de colores y tiempo de ejecución de la compresión. Cada métrica se utiliza entre la imagen original y la imagen resultante luego de la descompresión para valores de $k \in [32, 64, 128, 256]$, cada resultado se obtuvo del promedio de 50 ejecuciones.

Es evidente que al incrementar el valor de k, el error representado por el MSE disminuye, lo que indica una mejor calidad en la imagen resultante de la descompresión. Sin embargo, es importante destacar que este aumento en la calidad viene a expensas de un incremento significativo en el tiempo de ejecución. Este incremento en el tiempo es particularmente notable al pasar de k=128 a k=256, lo que refleja una relación exponencial entre k y el tiempo de ejecución.

Es también interesante observar cómo el uso de los colores únicos de la imagen original resulta en tiempos de ejecución más cortos en comparación con los casos donde se utilizan más colores para la compresión.

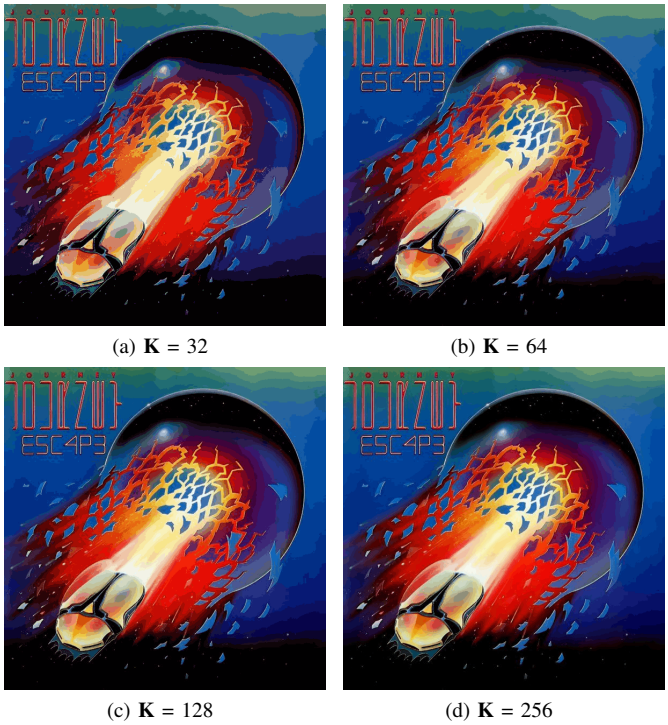
En general, los resultados de la tabla ofrecen información valiosa sobre el equilibrio entre calidad de compresión y eficiencia en función del valor de k, y proporcionan una base sólida para tomar decisiones informadas sobre la configuración óptima del algoritmo K-means en el contexto de la compresión de imágenes.

K	MSE	SSIM	CC (%)	CT (%)	τ_c (Seg)
Journey (158446 Colores Únicos)					
32	147.6	0.76	0.0202	21	0.30
64	84.41	0.82	0.0404	25	0.48
128	49.87	0.86	0.0808	29	0.84
256	30.22	0.90	0.1616	33	1.65
The Outfield (97882 Colores Únicos)					
32	61.52	0.867	0.0327	21	0.171
64	35.63	0.9	0.0654	25	0.262
128	21.56	0.926	0.1308	29	0.46
256	13.27	0.948	0.2615	33	0.97
Lena (148279 Colores Únicos)					
32	41.14	0.854	0.0216	21	0.244
64	24.93	0.897	0.0432	25	0.41
128	16.0	0.927	0.0863	29	0.73
256	10.34	0.948	0.1726	33	1.43
Mandrill (230427 Colores Únicos)					
32	130.61	0.86	0.0139	21	0.37
64	81.26	0.9	0.0278	25	0.0.617
128	52.021	0.93	0.0555	29	1.12
256	33.19	0.95	0.111	33	2.23

TABLE I: Resultados Practicos

En términos del Porcentaje de Compresión de Tamaño (CT), es notable cómo a partir del valor de k se puede reducir el tamaño en bytes de manera significativa. Esto demuestra la eficiencia de representar la información mediante k centroides. Además, es impresionante cómo trabajar con menos del 0.3% de los colores originales de una imagen no implica una distorsión visual considerable. Esta observación indica que en ciertas aplicaciones, las imágenes a menudo tienen una sobreabundancia de información de colores que es imperceptible para el ojo humano.

En la Figura 2, se muestra la salida del algoritmo de descompresión. A partir de esta figura, se infiere que visualmente, a partir de k=64, resulta difícil notar la falta de colores en la imagen. Es importante resaltar que esta observación está fuertemente influenciada por el tipo de imagen y la cantidad original de colores presentes en ella.



(a) **K** = 32 (b) **K** = 64

(c) **K** = 128 (d) **K** = 256

Fig. 2: Journey Para distintos Valores de **K**.

VI. Conclusión

Esta implementación y análisis del algoritmo K-means para la compresión de colores ilustra cómo la elección de k centroides influye en la calidad, eficiencia y tiempo de compresión. Además, resalta cómo la percepción humana permite eliminar colores imperceptibles en ciertos contextos, contribuyendo a reducir el espacio digital ocupado por la imagen. En última instancia, se puede concluir con certeza que el algoritmo K-means es efectivo en la compresión de colores, aunque en algunos casos puede carecer de eficiencia. Encontrar un equilibrio entre tiempo y error es crucial y depende del uso específico de la compresión.

References

[1] Stephen Boyd, "Introduction to Linear Algebra."

[2] M. Emre Celebi, "Improving the Performance of K-Means for Color Quantization," Image and Vision Computing, vol. 29, 2 Jan 2011.

[3] L. Brun, A. Tr´emeau, Digital Color Imaging Handbook, CRC Press, 2002, Ch. Color Quantization, pp. 589–638.