

# UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

ESCUELA PROFESIONAL DE CIENCIA DE LA COMPUTACIÓN  
ESTRUCTURA DE DATOS AVANZADOS



---

## Laboratorio 6: Cuantización de Color utilizando Octree

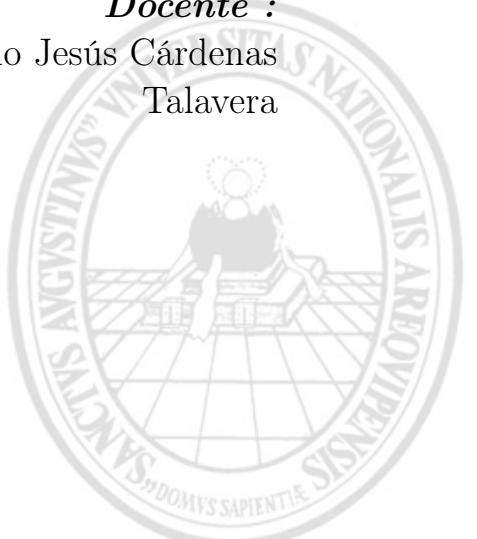
---

*Presentado por:*

Zavalaga Orozco, Rushell Vanessa  
Philco Puma, Josue Samuel

*Docente :*

Rolando Jesús Cárdenas  
Talavera



## 1. Introducción

La cuantización de color es una técnica ampliamente utilizada en procesamiento de imágenes para reducir la cantidad de colores en una imagen sin perder significativamente la calidad visual. Esto es especialmente útil en aplicaciones donde la capacidad de almacenamiento o los recursos de procesamiento son limitados, como en dispositivos embebidos, transmisión de imágenes en tiempo real, y compresión de imágenes.

El propósito de este proyecto es implementar un algoritmo basado en la estructura de datos **Octree** para realizar cuantización de color. Este algoritmo permite reducir el número de colores en una imagen a un conjunto definido, como 256 o 64 colores, mientras se genera una paleta de colores representativa. Los objetivos específicos son:

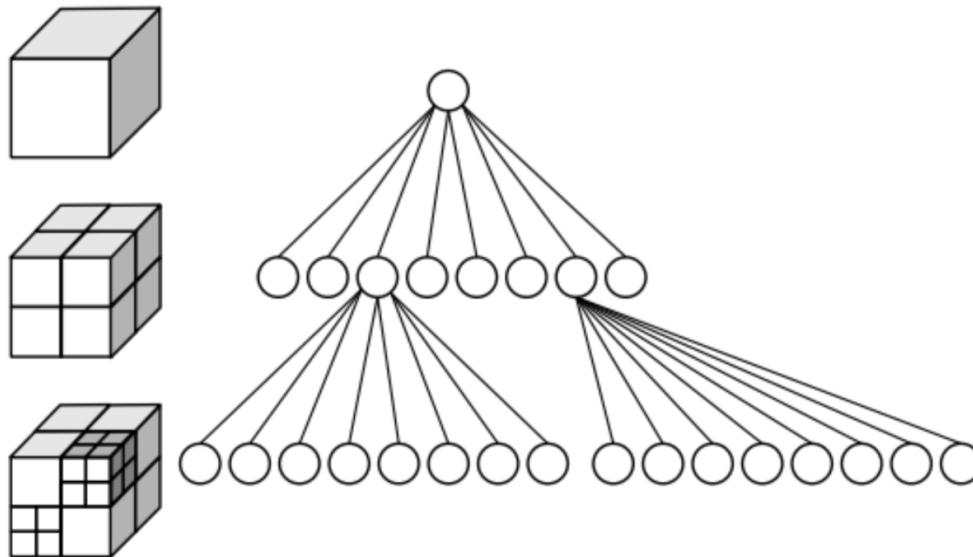
- Comprender el funcionamiento de la estructura **Octree** y su aplicación en procesamiento de imágenes.
- Desarrollar un algoritmo eficiente para la reducción de colores utilizando Octree.
- Implementar el algoritmo en un lenguaje de programación como C++ o Python y analizar su rendimiento.

Esta técnica es especialmente valiosa en aplicaciones como la edición de imágenes, compresión de datos y visualización en dispositivos con limitaciones de hardware.

## 2. Marco Teórico

### 2.1. Octree

En un OcTree, cada nodo subdivide el espacio que representa en ocho octantes (Fig. 1). En una región punto (PR) OcTree, el nodo almacena un punto tridimensional explícito, el cual es el "centro" de la subdivisión para ese nodo; el punto que define una de las esquinas para cada uno de los ocho hijos.



En una OcTree MX, el punto de subdivisión es implicitamente el centro del espacio que el nodo representa. El nodo raíz de una PR OcTree puede representar un espacio infinito; el nodo raíz de una OcTree MX debe representar un espacio con límite finito para que los centros implícitos estén bien definidos.

Las grillas OcTree nunca se consideran como un 'árbol kd', ya que los árboles kd dividen en una dimensión mientras que las grillas OcTree dividen alrededor de un punto. Los árboles kd además son siempre binarios, lo cual no se cumple para las grillas OcTree.

## 2.2. Color Quantization

La cuantificación del color reduce el número de colores usados en una imagen; esto es importante para visualizar imágenes en dispositivos que soportan un número limitado de colores y para eficiencia de compresión de ciertos tipos de imágenes. La mayoría de los editores de imágenes y muchos sistemas operativos soportan de forma nativa la cuantificación del color.



### 3. Descripción del Algoritmo

El algoritmo de cuantización de color utilizando un **Octree** divide recursivamente el espacio de colores RGB en ocho regiones cúbicas hasta que se alcanza un nivel de granularidad definido. Cada nodo del Octree representa un subconjunto del espacio de colores y puede almacenar información relevante, como el color promedio y la cantidad de píxeles asignados a ese nodo.

#### 3.1. Algoritmo de adición de un nuevo color

El proceso de adición de un nuevo color al Octree se realiza siguiendo los pasos:

1. **Inicialización:** Se recibe el color en formato RGB (Red, Green, Blue) como entrada.
2. **Navegación por el árbol:**
  - Comienza desde el nodo raíz, que representa todo el espacio de colores.
  - A cada nivel, se selecciona uno de los ocho hijos basado en los bits más significativos de los componentes del color (R, G, B).
3. **Inserción:** Si se llega al nivel de profundidad máximo, el nodo actual se marca como una hoja y se actualiza la información del color promedio y el número de píxeles representados.
4. **Creación de nuevos nodos:** Si el nodo correspondiente no existe, se crea un nuevo nodo hijo.

Este algoritmo permite construir el Octree de manera eficiente para representar el espacio de colores de la imagen.

### 3.2. Algoritmos de reducción de colores

El algoritmo de reducción de colores consiste en limitar el número total de nodos hoja en el Octree a un valor predefinido (por ejemplo, 256 o 64). Esto se logra mediante la fusión de nodos hoja:

1. **Recopilación de hojas:** Se recorre el Octree para identificar todos los nodos hoja actuales.
2. **Ordenamiento:** Se ordenan las hojas en función de su importancia, basada en criterios como el número de píxeles representados por cada hoja.
3. **Fusión de hojas:** Las hojas menos importantes se combinan con sus nodos padres, actualizando el color promedio y el número de píxeles representados.
4. **Actualización del árbol:** Después de cada fusión, se actualiza la estructura del Octree para reflejar los cambios.

Este proceso asegura que el Octree final tenga el número deseado de colores representados.

### 3.3. Algoritmo de construcción de paleta

Una vez que el Octree está optimizado, se construye la paleta de colores extrayendo los colores promedio de los nodos hoja. Los pasos son:

1. **Recorrido del árbol:** Se realiza un recorrido por los nodos hoja del Octree.
2. **Extracción de colores:** Se toma el color promedio almacenado en cada hoja.
3. **Generación de la paleta:** Los colores extraídos se almacenan en una lista o matriz que representa la paleta final.

Esta paleta puede utilizarse para mapear los colores originales de la imagen a su versión cuantizada, generando una imagen reducida en colores pero visualmente similar.

## 4. Implementación

El algoritmo de cuantización de colores basado en Octree fue implementado en Python. La estructura principal del código incluye tres clases fundamentales: `OctreeNode`, `Color`, y `OctreeQuantizer`. Estas clases trabajan juntas para construir el Octree, insertar colores, y generar una paleta de colores optimizada.

## 4.1. Descripción de la Implementación

El algoritmo se basa en subdividir el espacio de colores RGB utilizando una estructura de árbol Octree. Cada nodo del Octree almacena información sobre el promedio de color y la cantidad de píxeles representados. Los pasos principales son:

- Construcción del árbol: Los colores de la imagen son insertados nivel por nivel en el Octree.
- Reducción de colores: Se limitan los nodos hoja para reducir la cantidad total de colores.
- Generación de la paleta: Los colores promedio de los nodos hoja se extraen para crear la paleta final.

## 4.2. Fragmentos de Código

A continuación, se presentan fragmentos del código que destacan las partes clave de la implementación:

**Clase Color** Esta clase representa un color en formato RGB, con un atributo opcional para alfa:

```
class Color(object):
    def __init__(self, red=0, green=0, blue=0, alpha=None):
        self.red = red
        self.green = green
        self.blue = blue
        self.alpha = alpha
```

Listing 1: Clase Color para manejar colores RGB

**Clase OctreeNode** Un nodo del Octree almacena la información de color y apunta a sus hijos:

```
class OctreeNode(object):
    def __init__(self, level, parent):
        self.color = Color(0, 0, 0)
        self.pixel_count = 0
        self.palette_index = 0
        self.children = [None for _ in range(8)]
        if level < OctreeQuantizer.MAX_DEPTH - 1:
            parent.add_level_node(level, self)

    def is_leaf(self):
        return self.pixel_count > 0

    def add_color(self, color, level, parent):
        if level >= OctreeQuantizer.MAX_DEPTH:
            self.color.red += color.red
            self.color.green += color.green
```

```

    self.color.blue += color.blue
    self.pixel_count += 1
    return
index = self.get_color_index_for_level(color, level)
if not self.children[index]:
    self.children[index] = OctreeNode(level, parent)
self.children[index].add_color(color, level + 1, parent)
  
```

Listing 2: Clase para el nodo Octree

**Generación de la Paleta** La paleta de colores se construye a partir de los nodos hoja:

```

def generate_palette(self):
    palette = []
    for leaf in self.leaf_nodes:
        if leaf.pixel_count > 0:
            avg_red = leaf.color.red // leaf.pixel_count
            avg_green = leaf.color.green // leaf.pixel_count
            avg_blue = leaf.color.blue // leaf.pixel_count
            palette.append((avg_red, avg_green, avg_blue))
    return palette
  
```

Listing 3: Generación de la paleta de colores

## 5. Resultados

La implementación permite procesar imágenes para reducir sus colores a un número predefinido, como 256 o 64 colores, generando también una paleta visualmente representativa. A continuación, se presentan capturas de pantalla del proceso.

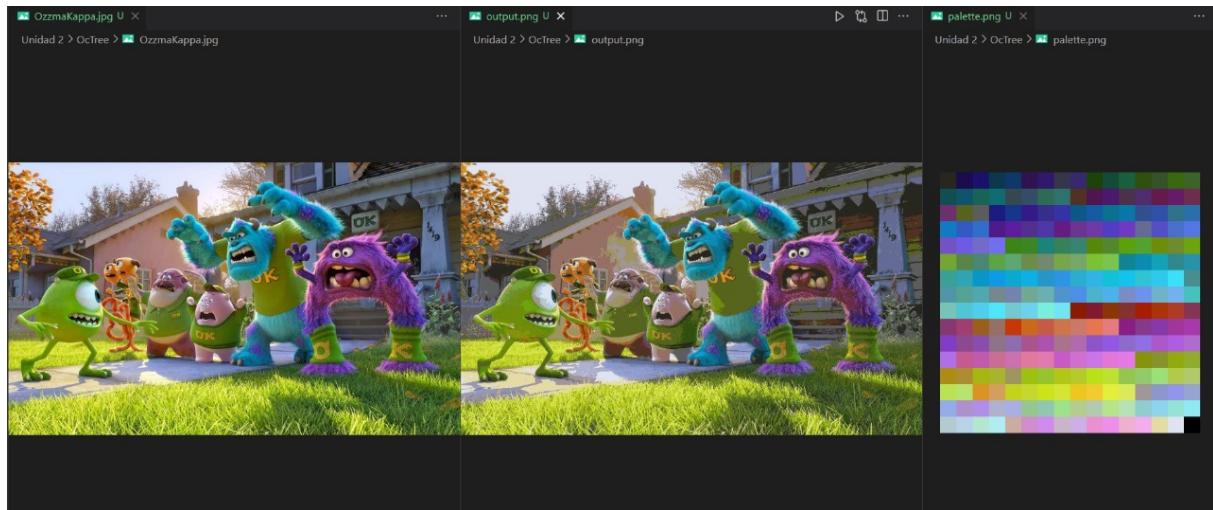


Figura 1: Izquierda: Imagen original. Centro: Imagen con reducción a 256 colores.  
 Derecha: Paleta

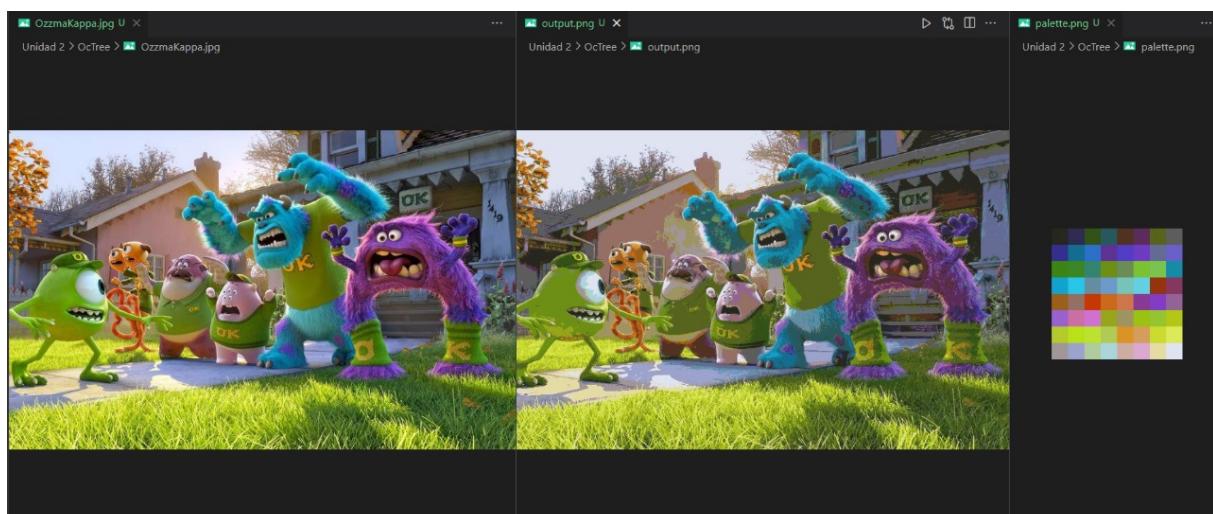


Figura 2: Izquierda: Imagen original. Centro: Imagen con reducción a 64 colores.  
 Derecha: Paleta

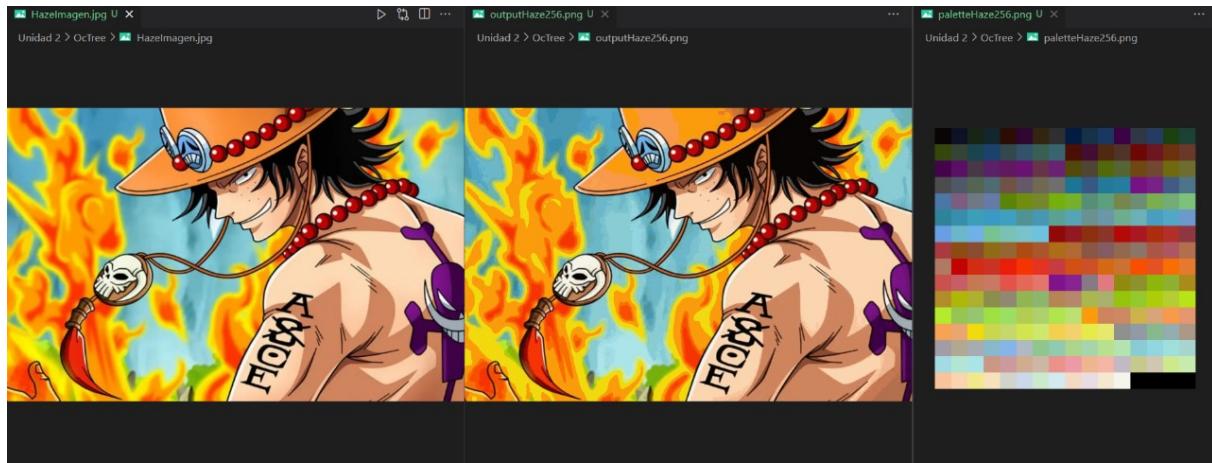


Figura 3: Izquierda: Imagen original. Centro: Imagen con reducción a 256 colores.  
Derecha: Paleta

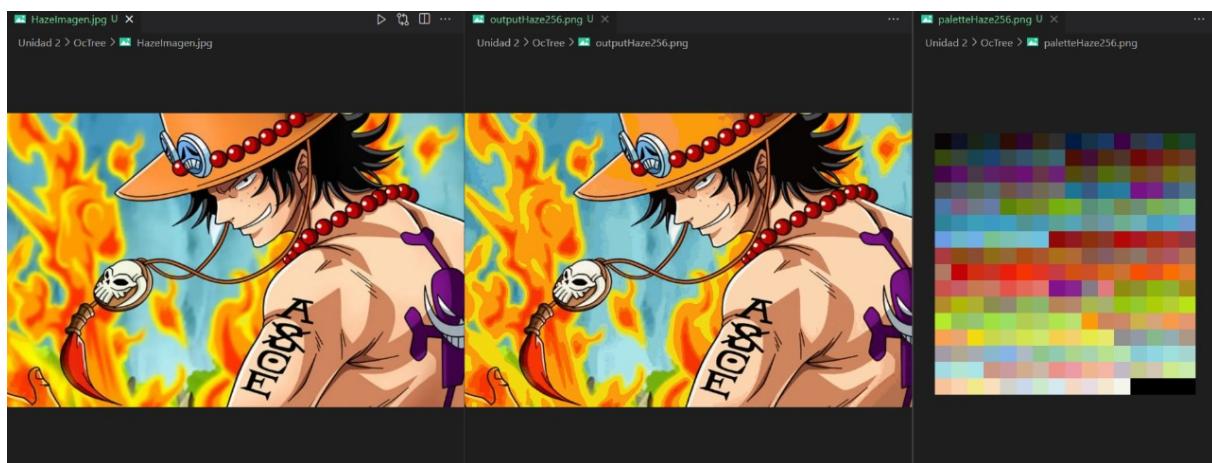


Figura 4: Izquierda: Imagen original. Centro: Imagen con reducción a 64 colores.  
Derecha: Paleta

## 6. Conclusiones

Este proyecto permitió explorar la implementación y el uso del Octree para la cuantización de colores, destacando tanto sus ventajas como sus desafíos.

### Hallazgos principales

- El uso de Octree es eficiente para reducir colores en imágenes, ya que permite una representación jerárquica del espacio RGB.
- La reducción de colores genera imágenes visualmente similares a las originales, mientras se optimiza el uso de memoria para almacenar colores.
- El algoritmo puede manejar imágenes de gran tamaño sin un impacto significativo en el rendimiento.

En resumen, el proyecto demostró la utilidad del Octree para tareas de procesamiento de imágenes y proporcionó una base sólida para futuras optimizaciones.