**LAB ACTIVITY**

**PENGOLAHAN CITRA DIGITAL**

**Pertemuan 11 – Kompresi Citra**

|  |  |
| --- | --- |
| Nama: Muhammad Rifqi Amir Putra | NPM: 062230701416 |
| Kelas: 5 CA | Mata Kuliah: Pengolahan Citra Digital |

**Alat dan Bahan:**

1. Text Editor
2. Python
3. Library Python numpy, opencv, matplotlib
4. Google Colab (Opsional)
5. **Kompresi Citra**
6. **Program**

**import numpy as np**

**import cv2 as cv**

**import heapq**

**from collections import defaultdict**

**import matplotlib.pyplot as plt**

**import networkx as nx**

**class HuffmanNode:**

**def \_\_init\_\_(*self*, *freq*, *symbol*=None, *left*=None, *right*=None):**

***self*.freq = *freq***

***self*.symbol = *symbol***

***self*.left = *left***

***self*.right = *right***

***self*.huff = ''**

**def \_\_lt\_\_(*self*, *other*):**

**return *self*.freq < *other*.freq**

**def calculate\_frequency(*data*):**

***# Menghitung frekuensi kemunculan setiap nilai pixel***

**freq\_dict = defaultdict(int)**

**for value in *data*:**

**freq\_dict[value] += 1**

**return freq\_dict**

**def build\_huffman\_tree(*freq\_dict*):**

***# Membangun pohon Huffman***

**heap = []**

**for symbol, freq in *freq\_dict*.items():**

**heapq.heappush(heap, HuffmanNode(freq, symbol))**

**while len(heap) > 1:**

**left = heapq.heappop(heap)**

**right = heapq.heappop(heap)**

**internal = HuffmanNode(left.freq + right.freq)**

**internal.left = left**

**internal.right = right**

**heapq.heappush(heap, internal)**

**return heap[0]**

**def generate\_codes(*root*, *code*='', *codes*=None):**

***# Menghasilkan kode Huffman untuk setiap simbol***

**if *codes* is None:**

***codes* = {}**

**if *root*.symbol is not None:**

***codes*[*root*.symbol] = *code***

**else:**

**generate\_codes(*root*.left, *code* + '0', *codes*)**

**generate\_codes(*root*.right, *code* + '1', *codes*)**

**return *codes***

**def compress\_image(*image\_path*):**

***# Melakukan kompresi citra menggunakan kode Huffman***

**img = cv.imread(*image\_path*)**

***# Proses untuk setiap channel (B, G, R)***

**compressed\_channels = []**

**huffman\_trees = []**

**codes\_per\_channel = []**

**for channel\_idx in range(3):**

**channel = img[:, :, channel\_idx].flatten()**

***# Hitung frekuensi***

**freq\_dict = calculate\_frequency(channel)**

***# Bangun pohon Huffman***

**root = build\_huffman\_tree(freq\_dict)**

**huffman\_trees.append(root)**

***# Generate kode Huffman***

**codes = generate\_codes(root)**

**codes\_per\_channel.append(codes)**

***# Kompresi data***

**compressed\_data = ''.join([codes[pixel] for pixel in channel])**

**compressed\_channels.append(compressed\_data)**

**return compressed\_channels, huffman\_trees, codes\_per\_channel, img.shape**

**def decompress\_image(*compressed\_channels*, *huffman\_trees*, *original\_shape*):**

***# Melakukan dekompresi citra***

**decompress\_channels = []**

**for compressed\_data, root in zip(*compressed\_channels*, *huffman\_trees*):**

***# Dekompresi channel***

**current = root**

**decompressed = []**

**for bit in compressed\_data:**

**if bit == '0':**

**current = current.left**

**else:**

**current = current.right**

**if current.symbol is not None:**

**decompressed.append(current.symbol)**

**current = root**

***# Reshape channel***

**decompress\_channel = np.array(decompressed).reshape(*original\_shape*[:2])**

**decompress\_channels.append(decompress\_channel)**

***# Gabungkan channels***

**decompress\_image = cv.merge(decompress\_channels)**

**return decompress\_image**

**def visualize\_huffman\_tree(*root*):**

***# Memvisualisasikan pohon Huffman menggunakan networkx***

**G = nx.Graph()**

**pos = {}**

**labels = {}**

**def add\_nodes(*node*, *x*=0, *y*=0, *layer*=1):**

**if *node* is None:**

**return**

***# Buat ID unik untuk node***

**node\_id = id(*node*)**

***# Tambahkan node ke graph***

**if *node*.symbol is not None:**

**G.add\_node(node\_id)**

**labels[node\_id] = f"{*node*.symbol}\n{*node*.freq}"**

**else:**

**G.add\_node(node\_id)**

**labels[node\_id] = str(*node*.freq)**

**pos[node\_id] = (*x*, *y*)**

***# Recursively add children***

**if *node*.left:**

**left\_id = id(*node*.left)**

**G.add\_edge(node\_id, left\_id)**

**add\_nodes(*node*.left, *x*-1/*layer*, *y*-1, *layer*+1)**

**if *node*.right:**

**right\_id = id(*node*.right)**

**G.add\_edge(node\_id, right\_id)**

**add\_nodes(*node*.right, *x*+1/*layer*, *y*-1, *layer*+1)**

**add\_nodes(*root*)**

**plt.figure(*figsize*=(12, 8))**

**nx.draw(G, *pos*=pos, *labels*=labels, *with\_labels*=True, *node\_color*='lightblue', *node\_size*=2000, *font\_size*=8, *font\_weight*='bold')**

**plt.title("Pohon Huffman")**

**plt.show()**

**def main(*image\_path*):**

***# Kompresi***

**compressed\_channels, huffman\_trees, codes, original\_shape = compress\_image(*image\_path*)**

***# Visualisasi pohon Huffman untuk channel pertama (Blue)***

**print("Visualisasi Pohon HUffman untuk Channel Blue:")**

**visualize\_huffman\_tree(huffman\_trees[0])**

***# Dekompresi***

**decompressed\_img = decompress\_image(compressed\_channels, huffman\_trees, original\_shape)**

***# Tampilkan hasil***

**plt.figure(*figsize*=(12, 4))**

**plt.subplot(121)**

**plt.imshow(cv.cvtColor(cv.imread(*image\_path*), cv.COLOR\_BGR2RGB))**

**plt.title("Gambar Asli")**

**plt.axis('off')**

**plt.subplot(122)**

**plt.imshow(cv.cvtColor(decompressed\_img, cv.COLOR\_BGR2RGB))**

**plt.title("Gambar Hasil Dekompresi")**

**plt.axis('off')**

**plt.show()**

***# Hitung dan tampilkan rasio kompresi***

**original\_size = cv.imread(*image\_path*).nbytes**

**compressed\_size = sum(len(channel) for channel in compressed\_channels) // 8**

**compression\_ratio = (1 - compressed\_size/original\_size) \* 100**

**print(f"\nUkuran file asli: {original\_size/1024:.2f} KB")**

**print(f"Ukuran file terkompresi: {compressed\_size/1024:.2f} KB")**

**print(f"Rasio kompresi: {compression\_ratio:.2f}%")**

**image\_path = 'Gambar/Pertemuan 10/White-Color-Abstract-Background-4k-Download.png'**

**main(image\_path)**

1. **Hasil**

**A diagram of a cloud

Description automatically generated with medium confidence**

**A blurry image of a couple of squares

Description automatically generated**

Ukuran file asli: 24300.00 KB

Ukuran file terkompresi: 17361.97 KB

Rasio kompresi: 28.55%