



IMAGE CODING AND COMPRESSION

Group 3

Edgrant H. S. (2206025016)

Giovan C. S. (2206816084)

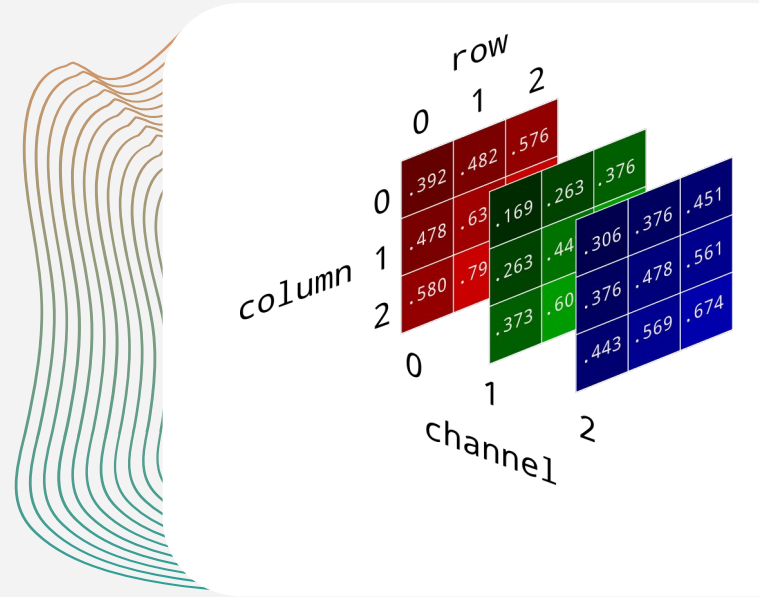
Nicholas S. (2206059396)



01 Introduction to Image Coding

Apa Itu Image Coding

Digital Image disimpan sebagai kumpulan piksel, masing-masing mewakili informasi warna dan kecerahan. Piksel-piksel ini disusun dalam kisi-kisi, di mana data setiap piksel didefinisikan menggunakan berbagai model warna seperti RGB (Red, Green, Blue) atau CMYK (Cyan, Magenta, Yellow, Black). Jumlah data yang diperlukan untuk menyimpan gambar tergantung pada resolusi dan kedalaman bit (jumlah bit yang digunakan untuk merepresentasikan setiap piksel). Gambar beresolusi tinggi dapat menghabiskan ruang penyimpanan yang signifikan, sehingga penyimpanan dan transmisi yang efisien menjadi tantangan.



Apa Itu Image Coding

Image coding atau biasa disebut dengan Image compression adalah pemrosesan digital image yang mengurangi ukuran file gambar sambil mempertahankan kualitas visual sebanyak mungkin. Kebutuhan akan kompresi Image compression muncul karena kendala penyimpanan, efisiensi transmisi, dan persyaratan Real Time image processing.



Mengapa Image Coding Penting



Peningkatan
efisiensi
penyimpanan
data

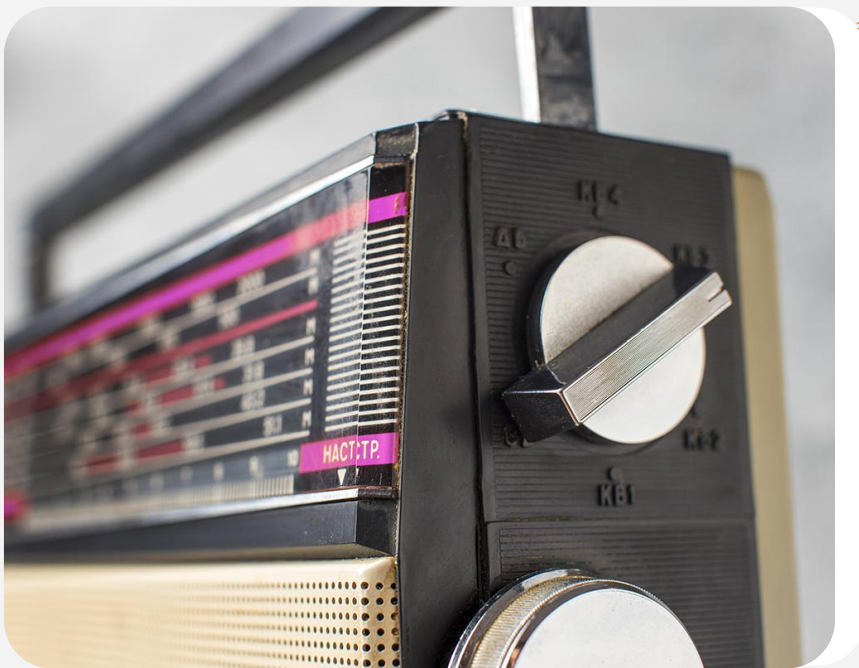


Peningkatan
transmisi dan
bandwidth



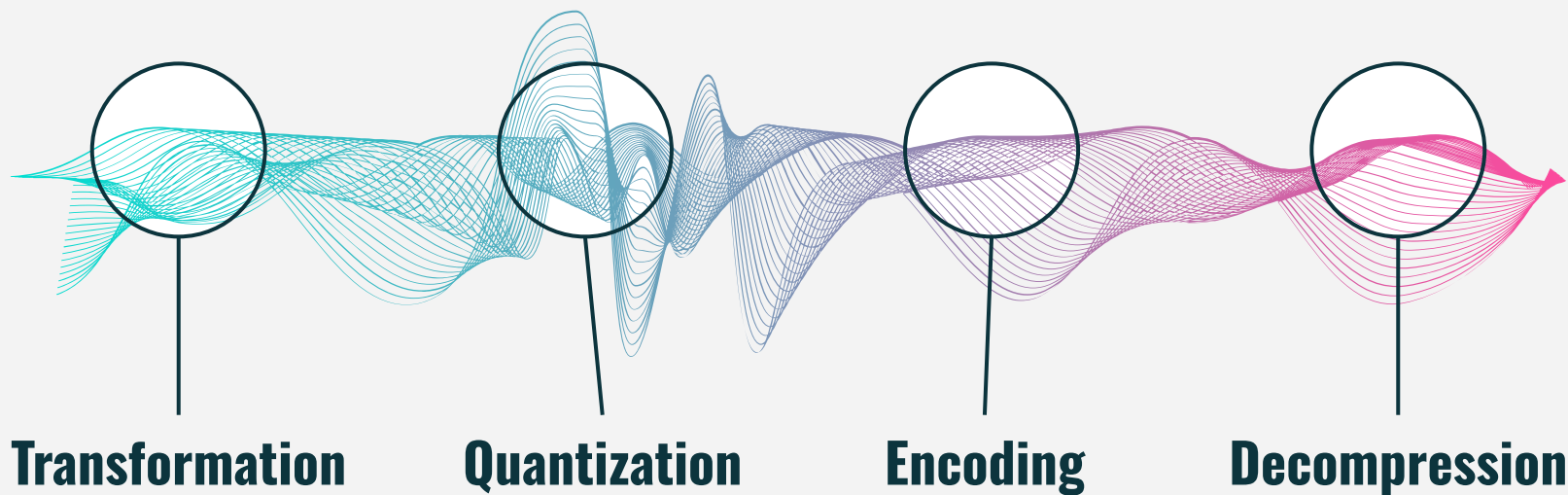
Kompatibilitas
untuk perangkat
dengan performa
terbatas



A decorative graphic consisting of several concentric, wavy lines in shades of pink and orange, flowing from the top left towards the bottom right, positioned between the radio image and the text.

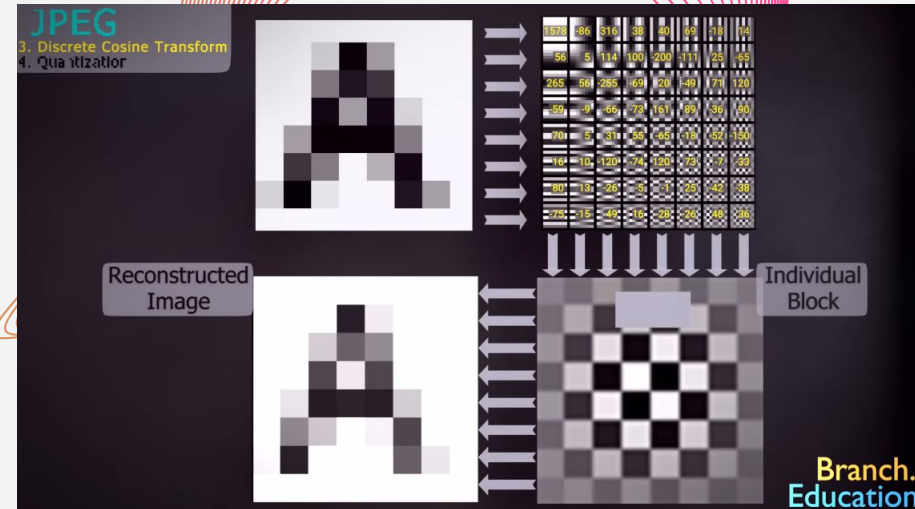
02 Details of Image Compression

Cara Kerja Image Coding



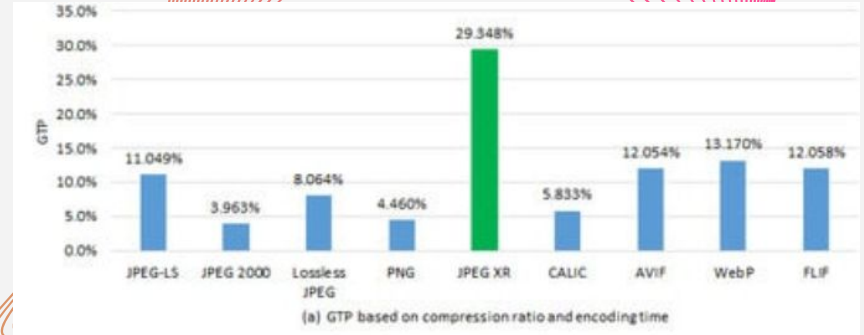
Transformation

Gambar ditransformasikan ke dalam domain yang berbeda, dari domain RGB / CMYK ke domain di mana kompresi lebih efisien. Teknik seperti Discrete Cosine Transform (DCT) untuk JPEG dan Wavelet Transform untuk JPEG 2000 mengubah data gambar spasial menjadi komponen frekuensi. Transformasi ini membantu memisahkan komponen frekuensi rendah, yang berisi sebagian besar detail penting gambar, dari komponen frekuensi tinggi, yang menyimpan detail dan noise yang lebih halus.



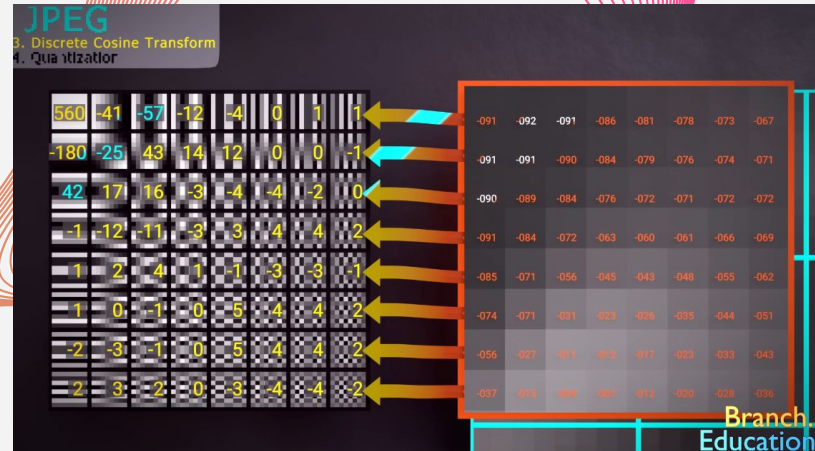
Transformation

Tingkat transformasi mempengaruhi efisiensi dari kompresi. Transformasi yang lebih tinggi memungkinkan kompresi yang lebih baik, tetapi hal tersebut dapat menimbulkan peningkatan kompleksitas dalam rekonstruksi.



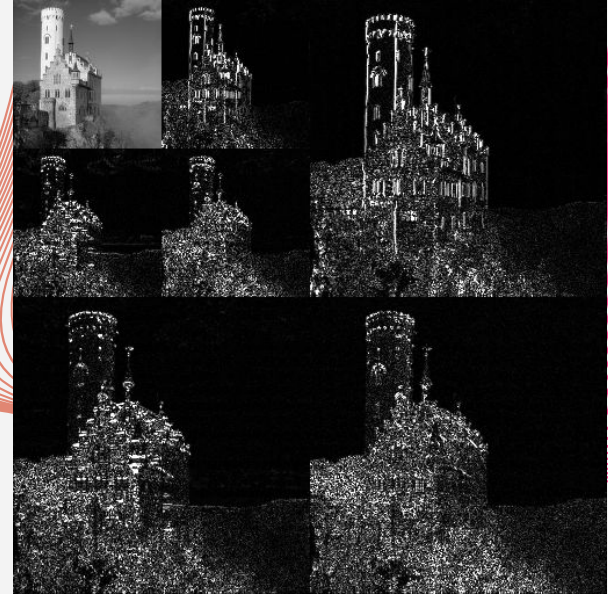
Transformation Method

- Discrete Cosine Transform (DCT): DCT mengubah gambar menjadi jumlah fungsi cosinus dengan frekuensi yang bervariasi. Hal ini membantu memisahkan informasi visual yang penting (komponen frekuensi rendah) dari detail yang kurang signifikan (komponen frekuensi tinggi), sehingga memungkinkan kompresi yang lebih baik dengan membuang detail yang kurang penting.



Transformation Method

- Wavelet Transform: Metode ini menguraikan gambar ke dalam skala atau resolusi yang berbeda. Daripada membagi gambar menjadi blok-blok ukuran tetap seperti DCT, Wavelet Transform menerapkan analisis multi-resolusi yang memecah gambar menjadi komponen-komponen yang semakin kecil dengan menerapkan filter low-pass untuk mengekstrak variasi yang halus (koefisien aproksimasi) dan filter high-pass untuk menangkap detail yang halus (koefisien detail)



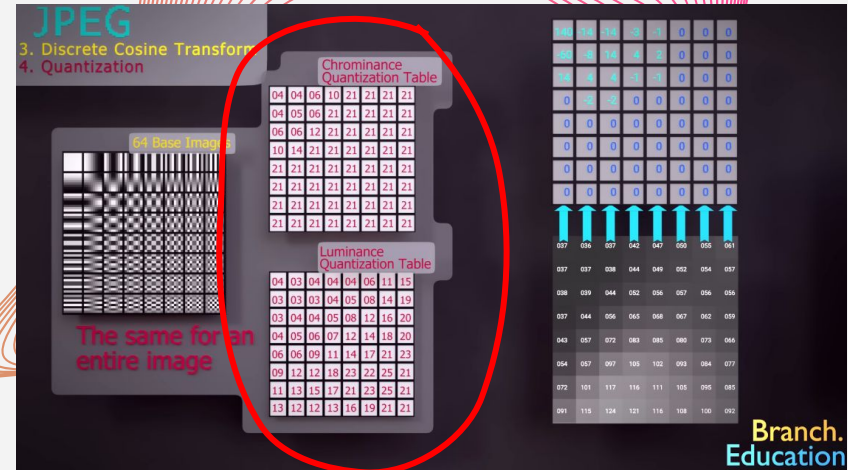
Quantization

Setelah transformasi, komponen frekuensi dikuantisasi, yang berarti nilai yang kurang signifikan dibulatkan atau dibuang. Komponen frekuensi rendah dipertahankan dengan presisi yang lebih tinggi, sementara komponen frekuensi tinggi (yang kurang berkontribusi pada persepsi visual) sering kali dikompresi dalam jumlah besar. Quantization secara signifikan mengurangi ukuran data dengan meminimalkan jumlah bit yang diperlukan untuk menyimpan detail yang kurang penting.



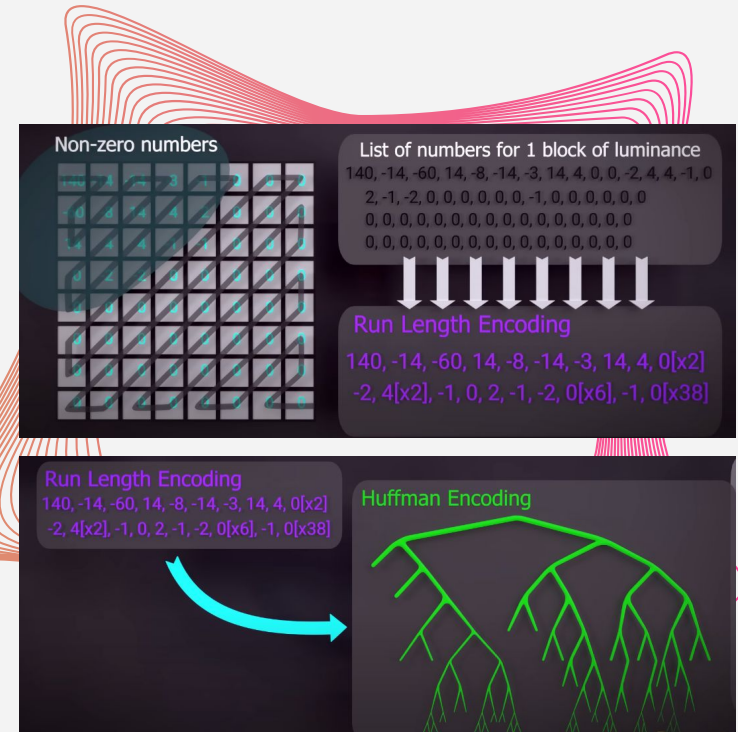
Quantization

Di sini terjadi pertukaran antara compression ratio dan image quality., Hal tersebut dikarenakan kuantisasi yang lebih tinggi menyebabkan rasio kompresi yang lebih besar, tetapi hal tersebut menyebabkan hilangnya kualitas gambar (keburaman, pikselasi, atau distorsi).



Encoding

Nilai - nilai yang telah dikuantisasi akan di-*encode* menggunakan algoritma kompresi seperti Huffman Encoding atau Arithmetic Encoding. Algoritma ini memberikan kode yang lebih pendek untuk nilai yang sering muncul dan kode yang lebih panjang untuk nilai yang kurang umum, sehingga mengurangi ukuran file secara keseluruhan. Encoding memastikan penyimpanan dan transmisi gambar yang efisien sambil mempertahankan informasi penting yang diperlukan untuk rekonstruksi. Pilihan metode encoding memiliki dampak pada kompresi akhir, dengan teknik yang lebih baik, gambar dapat disimpan dengan efisiensi yang lebih baik tanpa kehilangan kualitas yang signifikan.



Encoding Method

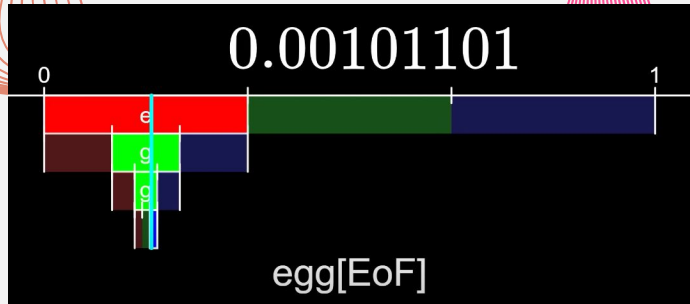
- **Huffman Coding:** Menetapkan kode biner yang lebih pendek untuk nilai yang lebih sering muncul dan kode yang lebih panjang untuk nilai yang lebih jarang muncul, sehingga mengurangi redundansi.
- **Arithmetic Coding:** Merepresentasikan seluruh aliran data sebagai angka pecahan tunggal antara 0 dan 1.
- **Run-Length Encoding (RLE):** Mengganti urutan nilai yang identik dengan pasangan count and value, sehingga mengurangi penyimpanan data yang berulang.
- **Lempel-Ziv-Welch (LZW) Encoding:** Membuat dictionary untuk urutan bit yang sering muncul dan menggantinya dengan kode yang lebih pendek

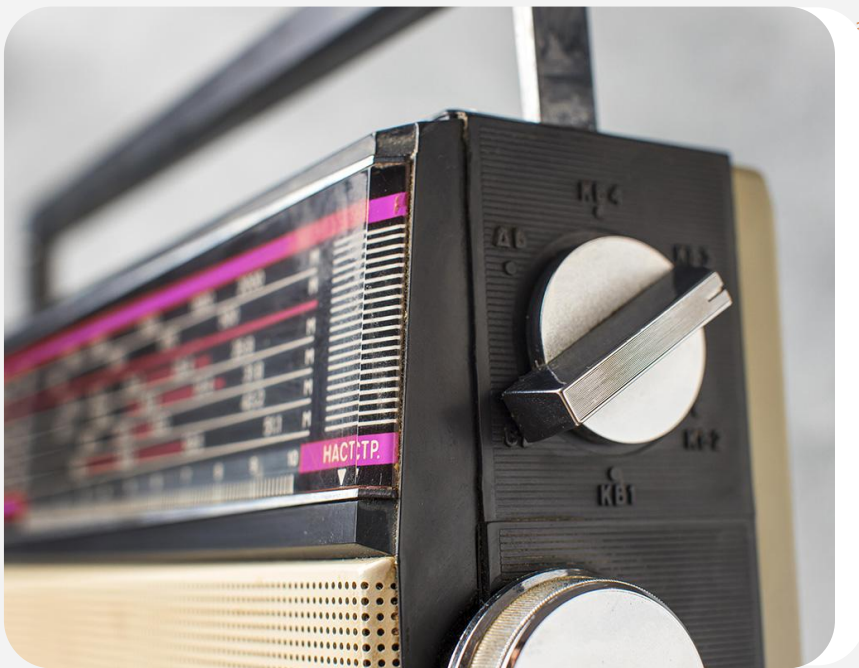
Huffman

Symbol:	' '	'I'	'W'	'E'	'R'	'D'	'H'	'S'	'B'	'A'
Probability:	1/4	1/5	1/10	1/10	1/10	1/20	1/20	1/20	1/20	1/20
Huffman code:	10	01	001	1110	1111	1100	0001	0000	11011	11010

I WISH I WERE A BIRD

Arithmetic




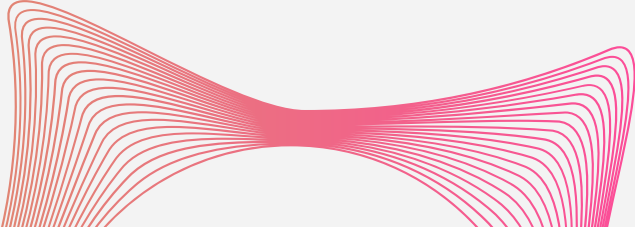


02

Types of Image Compression

Lossless Compression

Lossless compression bekerja dengan menghilangkan redundansi dalam data gambar tanpa membuang informasi apa pun, memastikan gambar asli dapat direkonstruksi dengan sempurna. Pada lossless compression, metode yang biasanya digunakan adalah Run-Length Encoding (RLE), Huffman Coding dan Lempel-Ziv-Welch (LZW). Lossless compression disimpan dalam format seperti PNG, GIF, dan TIFF. Lossless ideal digunakan untuk aplikasi yang memerlukan reproduksi gambar yang tepat.



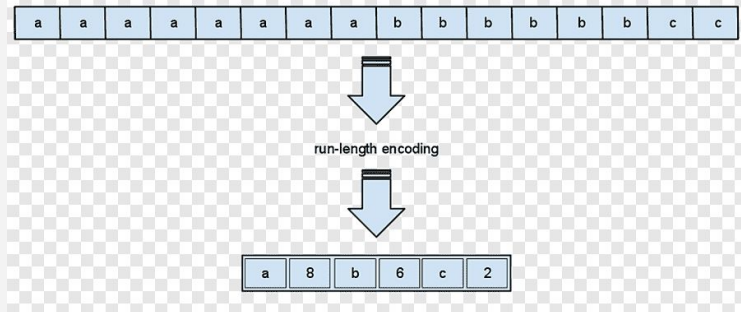
	BMP	JPG	PNG	GIF
Compressed	FALSE	TRUE	TRUE	TRUE
Lossless	TRUE	FALSE	TRUE	TRUE
Transparency	FALSE	FALSE	TRUE	TRUE
Translucency	FALSE	FALSE	TRUE	FALSE
Recommended for photographs	FALSE	TRUE	FALSE	FALSE
Recommended for static graphics/icons	FALSE	FALSE	TRUE	FALSE
Recommended for animated graphics/icons	FALSE	FALSE	FALSE	TRUE

Lossless Format

- PNG (Portable Network Graphics)

PNG menggunakan kompresi DEFLATE lossless (kombinasi dari LZ77 dan Huffman encoding). PNG mendukung transparansi melalui channel alpha. Format ini ideal untuk grafik web, logo, dan gambar yang membutuhkan tepi yang tajam.

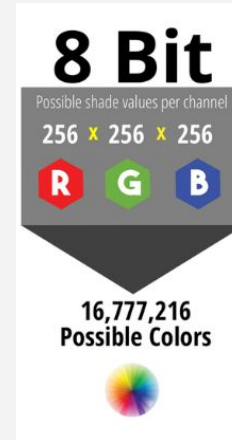
Algoritma Run Length Encoding yang digunakan PNG



Lossless Format

- GIF (Graphics Interchange Format)

GIF menggunakan kompresi LZW untuk mengurangi ukuran file tanpa kehilangan. GIF mendukung palet warna terbatas (maksimal 256 warna). Format ini biasanya digunakan untuk animasi dan ikon sederhana.



[103, 101, 101, 107, 105, 102, 105, 99, 256, 258, 260, 262]

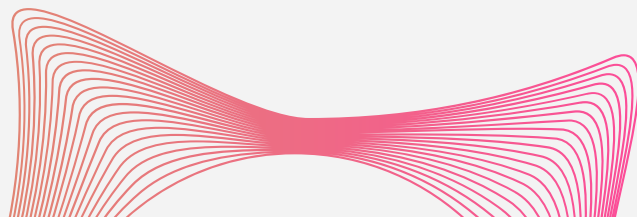
ge + ek

this allows us to encode and decode **without the need to provide the decoder with the dictionary created**

-	45
c	99
e	101
f	102
g	103
i	105
k	107
...	...
ge	256
ek	258
...	...
gee	265

Lossy Compression

Lossy compression mengurangi ukuran file dengan cara menghapus detail gambar yang kurang terlihat, terutama melalui kuantisasi dan pengkodean transformasi. Pada lossy compression biasanya digunakan metode DCT dan subsampling. Lossy compression disimpan dalam format seperti, JPEG, WebP, dan HEIC. Lossy ideal digunakan untuk menyeimbangkan kualitas dan ukuran file untuk mengoptimalkan penyimpanan dan transmisi.



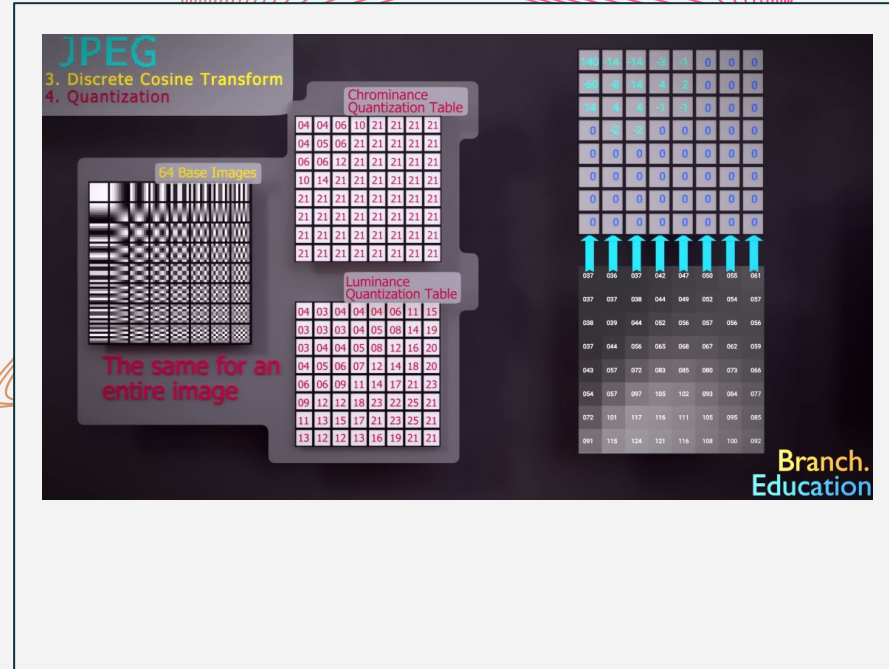
Summary Comparison:

Format	Compression	Lossy/Lossless	Transparency	Typical Use	Pros	Cons
JPEG	Lossy	Lossy	No	Photos, web images	Small file size, fast loading	Lossy, no transparency
PNG	Lossless	Lossless	Yes	Web graphics, logos, icons	High quality, transparency	Larger file size
HEIC	Lossy/Lossless	Lossy or Lossless	Yes	iPhone, mobile devices	Better compression, small file size	Limited support
GIF	Lossless	Lossless	Limited (1-bit)	Animations, simple graphics	Small file size, animation support	Limited colors (256 max)
TIFF	Lossless/Lossy	Lossless or Lossy	Yes (with alpha)	Professional imaging, archiving	High-quality, flexible	Large file sizes, less web support

Lossy Format

- JPEG (Joint Photographic Experts Group)

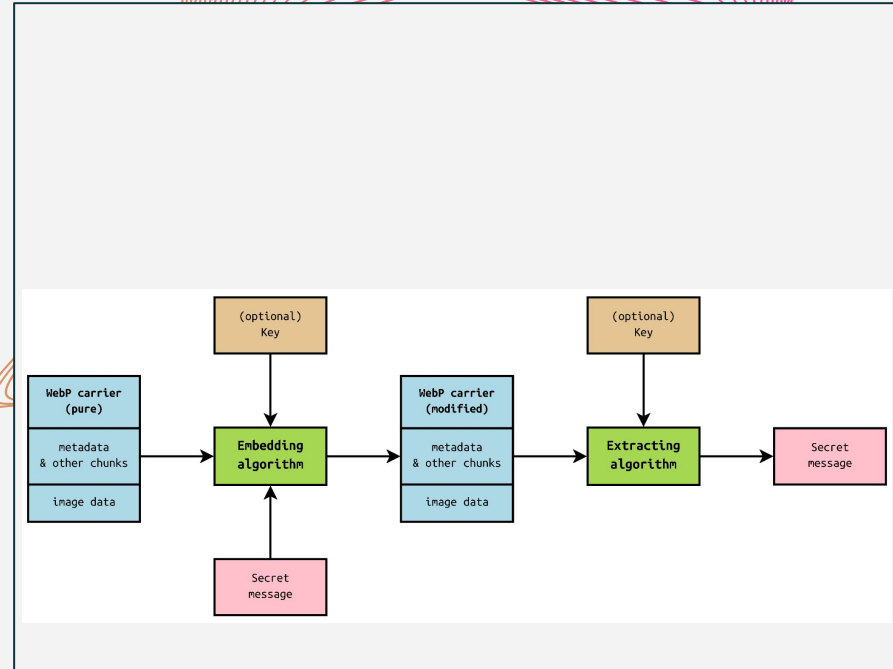
JPEG menggunakan DCT untuk mengubah data spasial menjadi komponen frekuensi dan menggunakan pengkodean Huffman atau aritmatika untuk pengkodean akhir. JPEG biasanya digunakan untuk foto dan gambar yang kompleks.



Lossy Format

- WebP

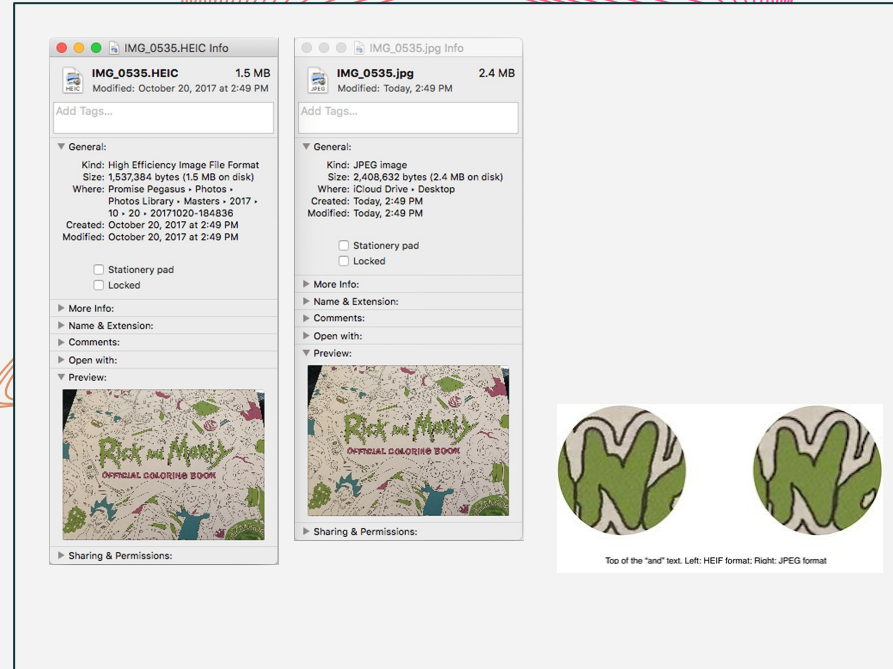
WebP Mendukung kompresi lossy (berbasis DCT) dan lossless (berbasis pengkodean entropi). Format ini memberikan rasio kualitas-ke-ukuran yang lebih baik daripada JPEG dan PNG. Format ini digunakan secara luas untuk pengoptimalan web.



Lossy Format

- HEIC (High Efficiency Image Coding)

HEIC dibuat Berdasarkan teknologi HEVC (High Efficiency Video Coding). HEIC menggunakan advanced intra-frame prediction and entropy coding. HEIC Menghasilkan kompresi yang superior dibandingkan dengan JPEG dengan retensi kualitas yang lebih baik. Format ini digunakan dalam ekosistem Apple untuk gambar berkualitas tinggi.





03

Demo

Code

```
%% Initialization
clear ; close all; clc

fprintf('\nRunning K-Means clustering on pixels from an image.\n\n');
```

Running K-Means clustering on pixels from an image.

```
% Load an image
A = double(imread('20250216_180310-small.jpg'));

A = A / 255; % Divide by 255 so that all values are in the range 0 - 1
```

```
% Size of the image
img_size = size(A);
```

```
% Reshape the image into an Nx3 matrix where N = number of pixels.
% Each row will contain the Red, Green and Blue pixel values
% This gives us our dataset matrix X that we will use K-Means on.
X = reshape(A, img_size(1) * img_size(2), 3);
```

Code

```
% Run your K-Means algorithm on this data
K = 16;
max_iters = 10;
```

```
% When using K-Means, it is important the initialize the centroids
% randomly.
initial_centroids = kMeansInitCentroids(X, K);
```

```
% Run K-Means
[centroids, idx] = runkMeans(X, initial_centroids, max_iters);
```

```
K-Means iteration 1/10...
K-Means iteration 2/10...
K-Means iteration 3/10...
K-Means iteration 4/10...
K-Means iteration 5/10...
K-Means iteration 6/10...
K-Means iteration 7/10...
K-Means iteration 8/10...
K-Means iteration 9/10...
K-Means iteration 10/10...
```

Code

```
fprintf('Program paused. Press enter to continue.\n');
```

Program paused. Press enter to continue.

```
pause;
```

```
fprintf('\nApplying K-Means to compress an image.\n\n');
```

Applying K-Means to compress an image.

```
% Find closest cluster members
```

```
idx = findClosestCentroids(X, centroids);
```

```
% Essentially, now we have represented the image X as in terms of the  
% indices in idx.
```

```
% We can now recover the image from the indices (idx) by mapping each pixel  
% (specified by its index in idx) to the centroid value
```

```
X_recovered = centroids(idx,:);
```

```
% Reshape the recovered image into proper dimensions
```

```
X_recovered = reshape(X_recovered, img_size(1), img_size(2), 3);
```

```
% Display the original image
```

```
subplot(1, 2, 1);
```

```
imagesc(A);
```

```
title('Original');
```

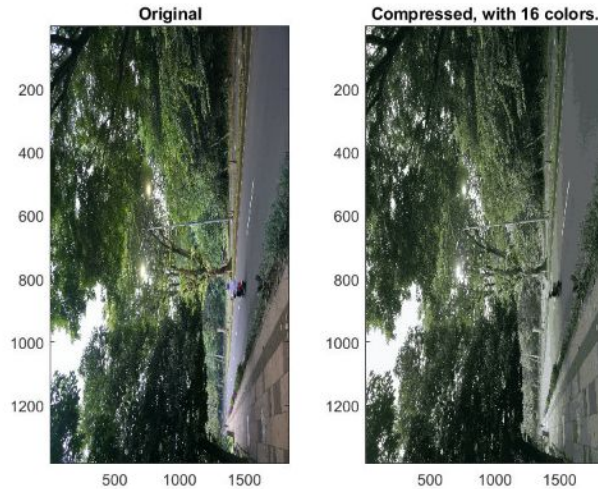
```
% Display compressed image side by side
```

```
subplot(1, 2, 2);
```

```
imagesc(X_recovered)
```

```
title(sprintf('Compressed, with %d colors.', K));
```

Hasil



```
% Save the compressed image
output_filename = '20250216_180310_compressed.jpg'; % Modify this as per your original filename
imwrite(X_recovered, output_filename);
fprintf('Compressed image saved as %s\n', output_filename);
```

Compressed image saved as 20250216_180310_compressed.jpg