

# Laporan: Implementasi K-Means Clustering pada Gambar

**Nama Anggota kelompok :**

Riski proba sadewo (31210191)  
Johanes Mula Febrian Sihombing (312210224)  
Faiz Dzaki Ramadhani (312210223)

## Pendahuluan

Clustering adalah teknik pengelompokan data yang bertujuan untuk mengelompokkan objek-objek yang memiliki karakteristik serupa ke dalam satu grup atau cluster. K-Means adalah salah satu algoritma clustering yang paling populer dan sering digunakan karena kesederhanaannya dan efisiensinya. Dalam laporan ini, kita akan menggunakan algoritma K-Means untuk segmentasi gambar, yang bertujuan untuk mengelompokkan piksel gambar ke dalam beberapa cluster berdasarkan kemiripan warnanya.

## Tujuan

Mengimplementasikan algoritma K-Means untuk segmentasi gambar.  
Menampilkan hasil segmentasi gambar dengan jumlah cluster yang ditentukan.

## Code

```
'''
import numpy as np
import matplotlib.pyplot as plt
import cv2

# Membaca gambar (gunakan path gambar yang sesuai)
image = cv2.imread('monarch.jpg')

# Memeriksa apakah gambar berhasil dimuat
if image is None:
    print("Error: Gambar tidak ditemukan atau tidak dapat dimuat.")
else:
    # Mengubah warna gambar menjadi RGB (dari BGR)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # Menampilkan gambar asli
    plt.imshow(image)
    plt.title('Original Image')
    plt.axis('off') # Menyembunyikan axis
    plt.show()

    # Mengubah bentuk gambar menjadi array 2D dari piksel dan 3 nilai warna (RGB)
    pixel_vals = image.reshape((-1, 3))

    # Mengonversi ke tipe float
    pixel_vals = np.float32(pixel_vals)

    # Langkah 1: Pilih jumlah cluster yang ingin dicari yaitu k.
```

k = 4

# Langkah 2: Tetapkan titik data secara acak ke salah satu k cluster.

# cv2.kmeans akan menangani penetapan acak secara internal

# Menentukan kriteria untuk algoritma berhenti berjalan, yang akan terjadi setelah 100 iterasi

# atau epsilon (akurasi yang dibutuhkan) tercapai pada 85%

criteria = (cv2.TERM\_CRITERIA\_EPS + cv2.TERM\_CRITERIA\_MAX\_ITER, 100, 0.85)

# Melakukan clustering k-means dengan jumlah cluster yang ditentukan sebagai k

# Pusat acak awalnya dipilih untuk clustering k-means

retval, labels, centers = cv2.kmeans(pixel\_vals, k, None, criteria, 10,  
cv2.KMEANS\_RANDOM\_CENTERS)

# Langkah 3: Kemudian hitung pusat clusternya.

# (cv2.kmeans menghitung pusat awal dan memperbaruinya dalam loop)

# centers sekarang berisi pusat dari setiap cluster

# Langkah 4: Hitung jarak titik data dari pusat masing-masing cluster.

# (cv2.kmeans menangani ini secara internal dengan menghitung jarak setiap titik ke pusat)

# Langkah 5: Bergantung pada jarak setiap titik data dari klaster, tetapkan kembali titik data ke klaster terdekat.

# (cv2.kmeans menetapkan kembali setiap titik ke pusat cluster terdekat)

# Langkah 6: Hitung lagi pusat cluster yang baru.

# (cv2.kmeans menghitung ulang pusat berdasarkan penetapan baru)

# Langkah 7: Ulangi langkah 4, 5 dan 6 hingga titik data tidak mengubah cluster, atau hingga mencapai jumlah iterasi yang ditetapkan.

# (cv2.kmeans mengulangi proses ini hingga konvergensi atau iterasi maksimum tercapai)

# Mengonversi data menjadi nilai 8-bit

centers = np.uint8(centers)

segmented\_data = centers[labels.flatten()]

# Mengubah bentuk data menjadi dimensi gambar asli

segmented\_image = segmented\_data.reshape((image.shape))

# Menampilkan gambar yang sudah di-segmentasi

plt.imshow(segmented\_image)

plt.title('Segmented Image with K-Means Clustering')

plt.axis('off') # Menyembunyikan axis


plt.show()

...

## Penjelasan Kode

### Langkah 1: Membaca Gambar

python


 Copy code

```
image = cv2.imread('monarch.jpg')
```

Langkah pertama adalah membaca gambar dari file menggunakan fungsi ``cv2.imread``. Gambar yang dibaca kemudian diubah warnanya dari BGR ke RGB menggunakan ``cv2.cvtColor`` untuk kompatibilitas dengan ``matplotlib``

## Langkah 2: Mengubah Warna Gambar

python


 Copy code

```
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

Gambar yang dibaca oleh OpenCV menggunakan format BGR secara default. Fungsi cv2.cvtColor digunakan untuk mengubah gambar menjadi format RGB.

## Langkah 3: Menampilkan Gambar Asli

python


 Copy code

```
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')
plt.show()
```

Gambar asli ditampilkan menggunakan ``matplotlib`` untuk visualisasi awal.

## Langkah 4: Mengubah Bentuk Gambar


python

 Copy code

```
pixel_vals = image.reshape((-1, 3))
```

Gambar diubah menjadi array 2D di mana setiap baris mewakili satu piksel dengan tiga nilai warna (RGB).


python

 Copy code

```
pixel_vals = np.float32(pixel_vals)
```

## Langkah 5: Mengonversi ke Tipe Float

python


 Copy code

```
pixel_vals = np.float32(pixel_vals)
```

Array piksel dikonversi ke tipe float untuk digunakan dalam algoritma K-Means.

## Langkah 6: Menentukan Kriteria K-Means

python


 Copy code

```
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 100, 0.85)
```

Kriteria untuk algoritma K-Means ditentukan, yaitu berhenti setelah 100 iterasi atau mencapai akurasi 85%.

### Langkah 7: Menjalankan K-Means Clustering

python


 Copy code

```
k = 4
retval, labels, centers = cv2.kmeans(pixel_vals, k, None, criteria, 10, cv2.KMEANS_RANDOM_
```

K-Means clustering dijalankan dengan jumlah cluster  $k$  yang ditentukan (dalam kasus ini, 4). Fungsi `cv2.kmeans` menangani proses penetapan titik data ke cluster, perhitungan pusat cluster, dan penyesuaian hingga konvergensi.

### Langkah 8: Mengonversi Data ke Nilai 8-Bit

python


 Copy code

```
centers = np.uint8(centers)
segmented_data = centers[labels.flatten()]
```

Pusat cluster dikonversi menjadi nilai 8-bit dan data gambar yang di-segmentasi dibentuk ulang menjadi dimensi gambar asli

### Langkah 9: Menampilkan Gambar yang Sudah Di-Segmentasi

python

 Copy code

```
segmented_image = segmented_data.reshape((image.shape))
plt.imshow(segmented_image)
plt.title('Segmented Image with K-Means Clustering')
plt.axis('off')
plt.show()
```

Gambar yang sudah di-segmentasi ditampilkan menggunakan matplotlib.

Hp	Itel p40
Dimension	1200x1600
Width	1200 pixels
Height	1600 pixels
Horizontal resolution	96 dpi
Vertical resolution	96 dpi
Bit depth	24



Hp	Samsung a54
Dimension	3060x4080
Width	3060 pixels
Height	4080 pixels
Horizontal resolution	96 dpi
Vertical resolution	96 dpi
Bit depth	24



Hp	Poco f4
Dimension	3123x4160
Width	3123 pixels
Height	4160 pixels
Horizontal resolution	96 dpi
Vertical resolution	96 dpi
Bit depth	24



## Kesimpulan

Algoritma K-Means adalah metode yang efektif untuk segmentasi gambar berdasarkan warna. Dengan mengelompokkan piksel ke dalam beberapa cluster, kita dapat menyederhanakan gambar dan menyoroti area-area dengan warna serupa. Implementasi di atas menunjukkan langkah-langkah dasar untuk menjalankan K-Means clustering pada gambar menggunakan OpenCV dan matplotlib.