

• •

>>> XỬ LÝ ẢNH

# HỆ THỐNG PHÂN ĐOẠN ẢNH BẰNG K-MEANS VÀ THRESHOLDING



• •

# THÀNH VIÊN NHÓM

1. PHẠM VĂN ĐỨC - B22DCCN243

## ĐỀ TÀI

### ĐỀ TÀI 03: HỆ THỐNG PHÂN ĐOẠN ẢNH BẰNG K-MEANS VÀ THRESHOLDING

- Nội dung: Phân đoạn ảnh (tách nền và đối tượng) bằng
  - Phân cụm K-means
  - Thuật toán Otsu.
- Nghiệm thu sản phẩm: Cho phép người dùng tải/lưu ảnh , xem kết quả trực quan.



# NỘI DUNG

01

**PHÂN ĐOẠN (PHÂN VÙNG) ẢNH**

02

**PHƯƠNG PHÁP PHÂN NGƯỠNG**

03

**PHÂN VÙNG ẢNH SỬ DỤNG  
THUẬT TOÁN KMEANS**

04

**DEMO KẾT QUẢ**

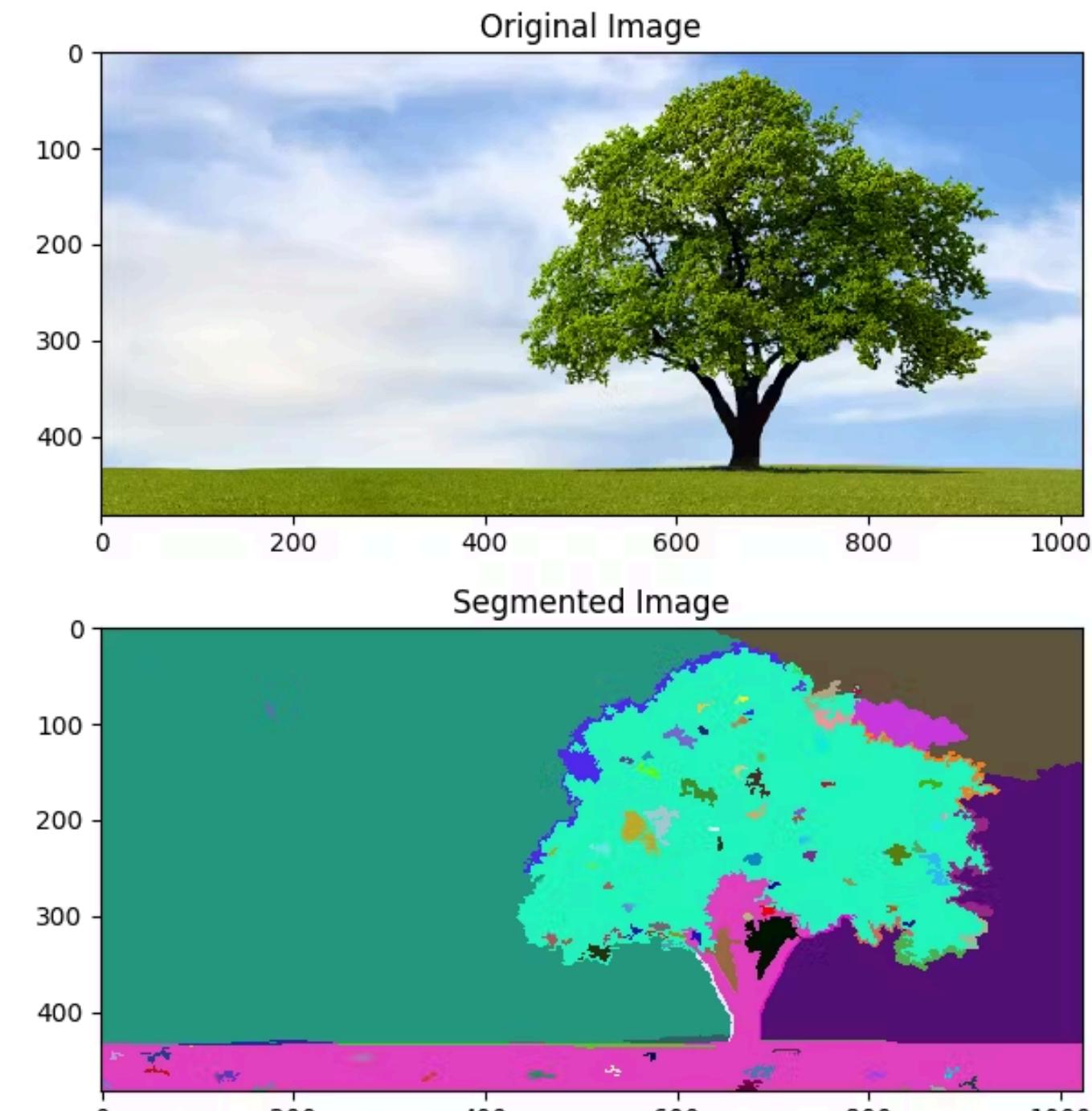


# BÀI TOÁN PHÂN ĐOẠN (PHÂN VÙNG) ẢNH

**Phân vùng ảnh** nhằm tới việc đạt được một biểu diễn phù hợp của hình ảnh để sử dụng cho quá trình xử lý tiếp theo

>>> Nói cách khác: Phân vùng ảnh **thường là bước tiền xử lý** nhằm phân tích đối tượng thành các biểu hiện đơn giản hơn (cạnh, góc, điểm, ...) cho các tác vụ tiếp theo trong thị giác máy tính.

- Trong nhận dạng đối tượng (Object recognition) => Thực hiện tách vùng vật thể trước khi đưa vào mô hình nhận dạng.
- Trong theo dõi đối tượng (Tracking) => Xác định vùng mục tiêu.
- Trong trích đặc trưng (Feature extraction)
- Trong đo đạc, phân tích
- ...



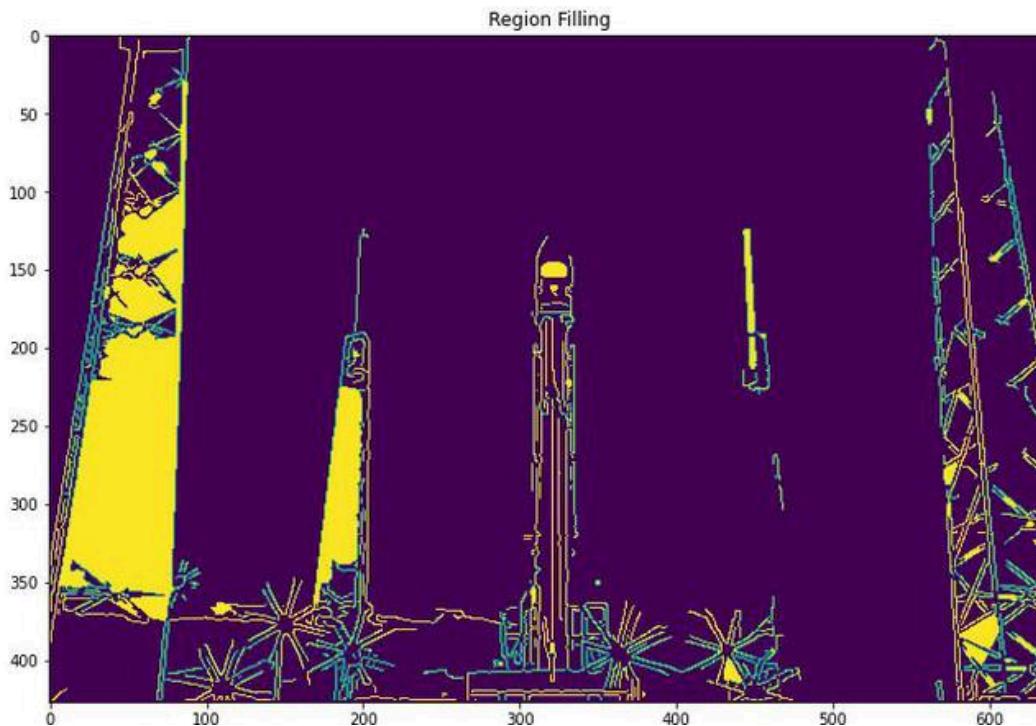
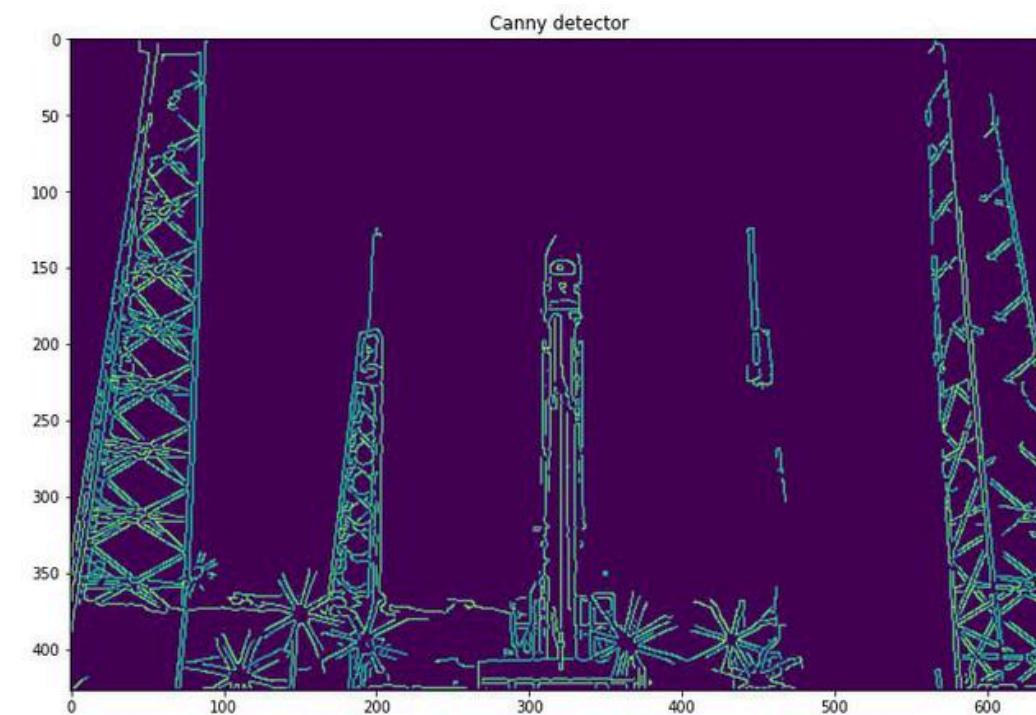
# BÀI TOÁN PHÂN ĐOẠN (PHÂN VÙNG) ẢNH

Phần lớn các thuật toán phân vùng đều dựa vào một trong hai đặc điểm cơ bản của giá trị cường độ:

- **Sự không liên tục** (discontinuity).
- **Tính tương đồng** (similarity)

>>> 02 cách tiếp cận để thực hiện phân đoạn ảnh:

1. **Phân vùng dựa vào biên**: dựa vào sự thay đổi đột ngột của mức xám (dựa trên **sự không liên tục**).
  - Phát hiện biên.
2. **Phân vùng dựa vào miền**: phân chia bức ảnh thành các vùng tương tự nhau theo một tập các tiêu chí cho trước (dựa trên **tính tương đồng**).
  - *Phân ngưỡng biên độ*.
  - *Phân vùng theo miền đồng nhất*.
  - Phân vùng theo kết cấu bề mặt.



# BÀI TOÁN PHÂN ĐOẠN (PHÂN VÙNG) ẢNH - PHÂN VÙNG DỰA VÀO MIỀN

Thực hiện chia ảnh R thành các vùng nhỏ hơn R1, R2,..., Rn.

Giả sử P(Ri) là một mệnh đề logic mà giá trị pixel trong vùng Ri thỏa mãn

>>>  $P(R_i) == \text{True}$

Các vùng nhỏ phải thỏa mãn các điều kiện sau:

- (1)  $\bigcup_{i=1}^n R_i = R$
- (2)  $R_i$  là một tập kết nối
- (3)  $R_i \cap R_j = \emptyset, \forall i, j \text{ and } i \neq j$
- (4)  $P(R_i) = \text{TRUE}, \forall i = 1, 2, \dots, n$
- (5)  $P(R_i \cup R_j) = \text{FALSE}$  với bất kì hai vùng liền kề

# PHƯƠNG PHÁP PHÂN NGƯỠNG

## PHÂN NGƯỠNG ĐƠN

$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T \\ 0 & \text{if } f(x, y) \leq T \end{cases}$$

Để tách các đối tượng khỏi nền ta chọn ngưỡng  $T$  sao cho nó phân tách được 2 miền này

*Phân ngưỡng toàn cục:* Phân ngưỡng đơn với  $T$  không đổi trên toàn bộ ảnh

## PHÂN NGƯỠNG KÉP

$$g(x, y) = \begin{cases} 0 & \text{if } f(x, y) \leq T_1 \\ 1 & \text{if } T_1 < f(x, y) \leq T_2 \\ 2 & \text{if } f(x, y) > T_2 \end{cases}$$

## VẤN ĐỀ CHỌN NGƯỠNG

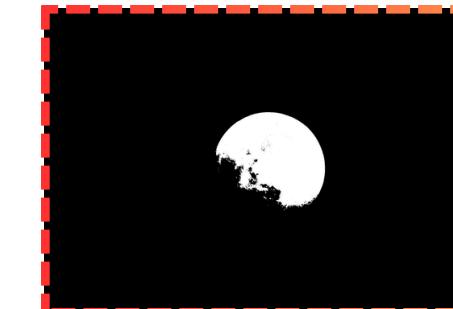
- Ngưỡng cố định: sử dụng ngưỡng được chọn độc lập với dữ liệu hình ảnh.
- Ngưỡng dựa vào biểu đồ: ngưỡng được chọn từ biểu đồ về cường độ sáng.

# PHƯƠNG PHÁP PHÂN NGƯỠNG -

## PHƯƠNG PHÁP OTSU (PHÂN NGƯỠNG TOÀN CỤC TỐI ƯU)

**Ý tưởng cơ bản:** Các nhóm được phân ngưỡng tốt phải khác biệt với nhau về giá trị cường độ của các pixel. Ngưỡng nào cho sự phân biệt tốt nhất giữa các nhóm (theo khía cạnh giá trị cường độ) sẽ được coi là ngưỡng tối ưu.

**“Tối ưu”:** theo hướng tối đa hóa phương sai giữa các nhóm



### THUẬT TOÁN OTSU

**INPUT:** image (grayscale)

**OUTPUT:** threshold  $T^*$

**1. Tính xác suất pixel -  $p[i]$**

//  $L = \text{grayLevel}(\text{image})$

// histogram  $h[i]$  cho  $i = 0..L-1$

// totalPixel =  $M*N$

$p[i] = h[i] / \text{total\_pixels}$  cho  $i = 0..L-1$

**2. Tính các tổng tích lũy -  $p_i[k]$**

for  $k = 1$  to  $L-1$ :

$p_i[k] = p[i] + p_i[k - 1]$

**3. Tổng trung bình tích lũy đến mức  $k$  -  $m[k]$**

for  $k = 1$  to  $L-1$ :

$m[k] = k*p_i[k] + m[k - 1]$

**4. Tính trung bình cường độ toàn cục -  $mG$**

$mG = m[L-1]$

**5. Tính phương sai giữa các nhóm -  $\sigma_B^2[k]$**

$\sigma_B^2 = -\inf$

$T = 0$

for  $t = 0$  to  $L-2$ :

$\sigma_B^2_t = \text{pow}(mG*p_i[t] - m[t], 2) / (p_i[t] * (1 - p_i[t]))$

if  $\sigma_B^2_t > \sigma_B^2$ :

$\sigma_B^2 = \sigma_B^2_t$

$T = t$

**6. Trả về ngưỡng tối ưu -  $T^*$**

return  $T$

$$P_1(k) = \sum_{i=0}^k p_i$$

$$m(k) = \sum_{i=0}^k i.p_i$$

$$\sigma_B^2(k) = \frac{[m_G P_1(k) - m(k)]^2}{P_1(k)[1 - P_1(k)]}$$

# PHÂN VÙNG ẢNH SỬ DỤNG THUẬT TOÁN KMEANS



# PHÂN VÙNG ẢNH SỬ DỤNG THUẬT TOÁN KMEANS

**Ý tưởng cơ bản:** phân chia một tập hợp các quan sát ( $Q$  - các điểm ảnh) thành một số cụm cụ thể ( $k$ ) => gán pixel cho cụm có **giá trị trung bình** (*nguyên mẫu/tâm cụm*) gần nhất.

- Tập Q quan sát:  $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_Q\}$
- Mỗi quan sát:  $\mathbf{x}_j = [z_{j,1}, z_{j,2}, \dots, z_{j,n}]^T$ , VD:  $\mathbf{x}_j = [r_j, g_j, b_j, x_j, y_j]^T$
- K phân cụm:  $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_K$
- Mỗi phân cụm:  $\mathcal{C}_k$ :  $\mathbf{m}_i = \frac{1}{\text{card}(\mathcal{C}_k)} \sum_{\mathbf{x}_j \in \mathcal{C}_k} \mathbf{x}_j$

Mục tiêu:

$$\{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_K\} = \arg \min_{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_K} \sum_{k=1}^K \sum_{\mathbf{x}_j \in \mathcal{C}_k} \|\mathbf{x}_j - \mathbf{m}_k\|^2$$

## THUẬT TOÁN KMEAN TIÊU CHUẨN

**INPUT:** image, số cụm  $K$

**OUTPUT:** segmentedImage

**1. Khởi tạo thuật toán:** Chỉ định  $k$ -clusters ( $k$  cụm) ban đầu.

$$\{C_1, C_2, \dots, C_k\}$$

euclidean=1e-4

**2. Lặp:**

2.1. Gán từng mẫu vào tập cụm có giá trị trung bình gần nhất:

$$\mathbf{x}_i \rightarrow \mathcal{C}_k \text{ if } \|\mathbf{x}_i - \mathbf{m}_k\|^2 < \|\mathbf{x}_i - \mathbf{m}_j\|^2, \forall j \neq k$$

2.2. Cập nhật các trung tâm cụm (giá trị trung bình):

$$\mathbf{m}_k = \frac{1}{\text{card}(\mathcal{C}_k)} \sum_{\mathbf{x}_j \in \mathcal{C}_k} \mathbf{x}_j, \forall k = 1, 2, \dots, K$$

2.3. Kiểm tra mức độ hoàn thành:

```
if distance(old_m, new_m) <= euclidean:
    break
```

**3. Tái tạo ảnh và trả về**

# PHÂN VÙNG ẢNH SỬ DỤNG THUẬT TOÁN KMEANS++

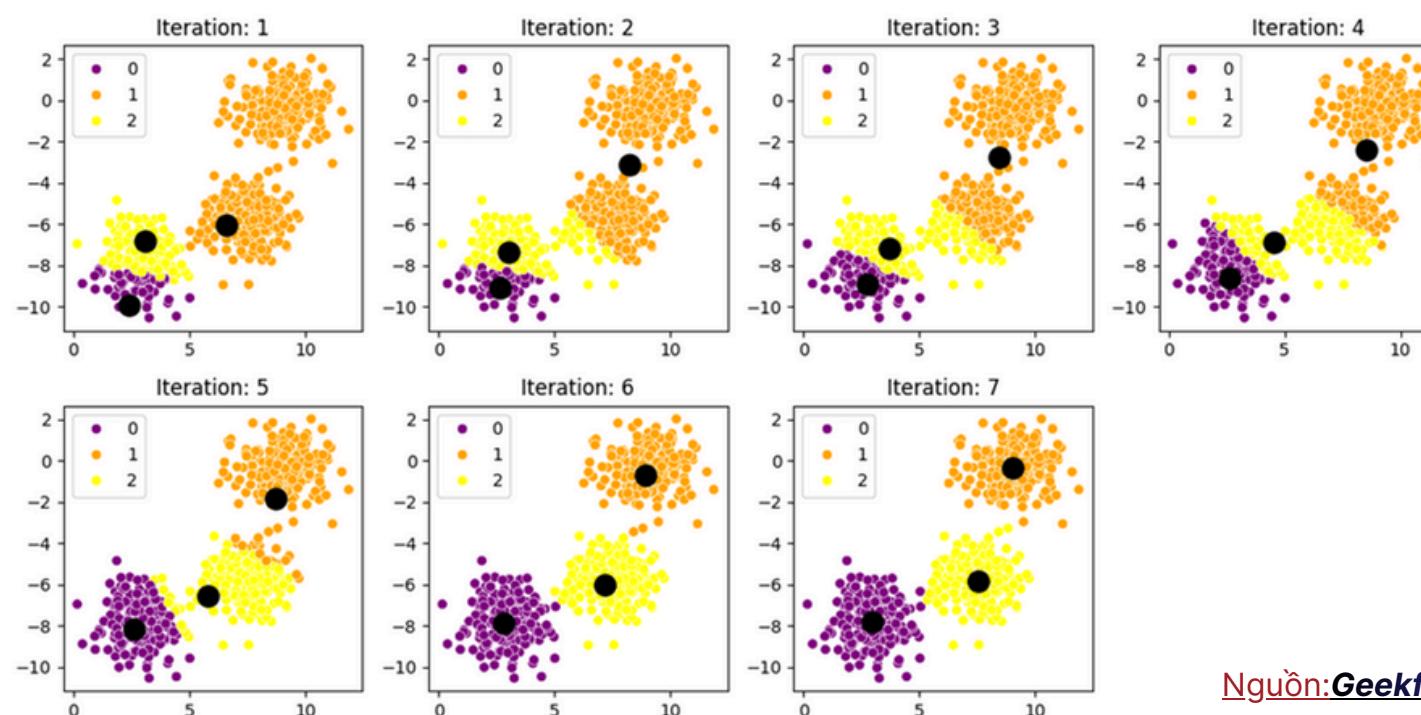
## THUẬT TOÁN KMEANS TIÊU CHUẨN

Chọn K cluster ban đầu:

- Chọn ngẫu nhiên.
- Phân bổ đều trên tập mức xám.
- ...

=> Vấn đề: Tâm cụm ban đầu ảnh hưởng tới kết quả thuật toán

- Các cụm quá gần nhau.
- Kết quả không ổn định.



Nguồn: [Geekforgeeks](#)

## THUẬT TOÁN KMEANS++

Chọn cluster khởi tạo thứ i - Ci dựa trên khoảng cách với các cluster trước đó: C1, C2, ..., Ci-1. Cụ thể:

1. Tính khoảng cách tối thiểu tới cụm gần nhất.

$$D2min[k] = \min(\text{pow}(\text{distance}(g[k], C[j]), 2))$$

$k=0,1,2,\dots,L-1; j=1,2,\dots,i-1$

2. Tính khoảng cách tổng thể:

$$W[k] = D2min[k]^*h[k], k=0,1,2,\dots,L-1$$

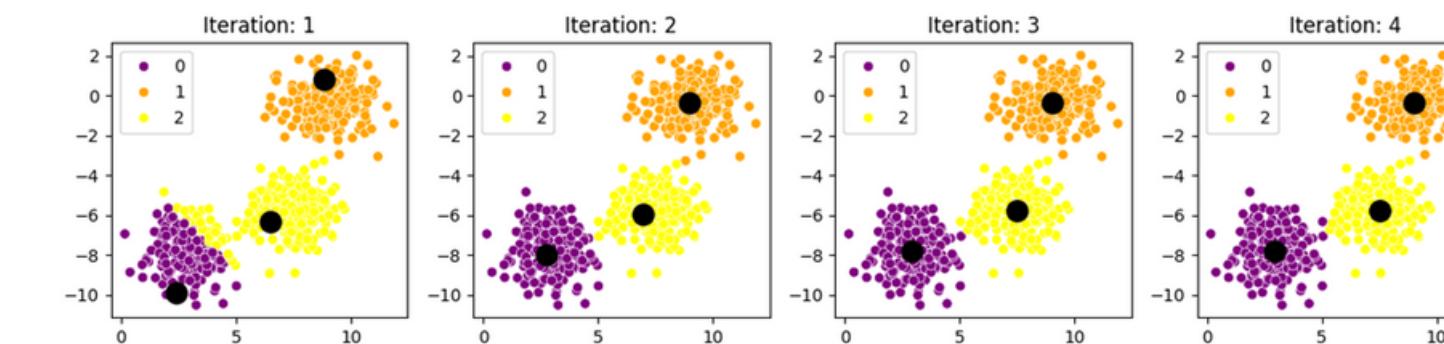
3. Sinh xác suất tích lũy:

$$P[k] = P[k-1] + W[k]/\text{sum}(W) // \text{ví dụ: } P = [10, 50, 70, 90, 100]$$

4. Random p và chọn cụm tiếp theo

// ví dụ: p=65 => P thuộc khoảng 50-70 (P[2])

// => 2 được chọn làm cụm



# PHÂN VÙNG ẢNH SỬ DỤNG THUẬT TOÁN KMEANS++



THUẬT TOÁN KMEANS TIÊU CHUẨN,  $k=3$ ,  $euclidean=1e-10$

# PHÂN VÙNG ẢNH SỬ DỤNG THUẬT TOÁN KMEANS++



THUẬT TOÁN KMEANS++,

$k=3$ ,  $\text{euclide}=1e-10$

# PHÂN VÙNG ẢNH SỬ DỤNG THUẬT TOÁN KMEANS++

Ảnh hưởng của số cụm ( $k$ -Cluster) và hệ số Euclidean

# PHÂN VÙNG ẢNH SỬ DỤNG THUẬT TOÁN KMEANS++



Ảnh gốc

K = 2



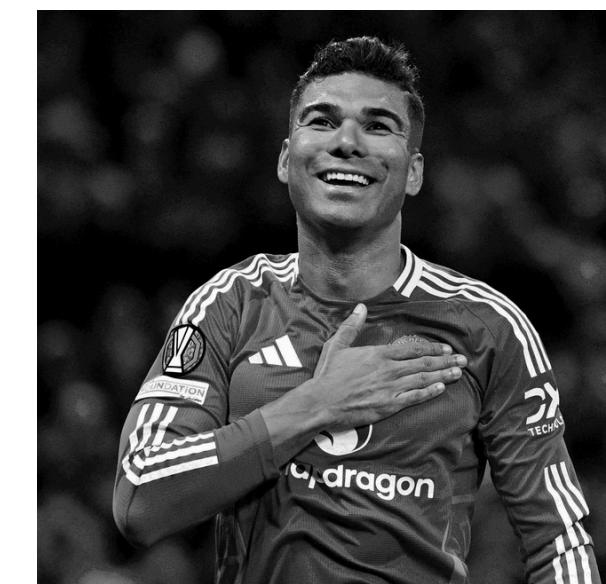
K = 5



K = 10



K = 25



# PHÂN VÙNG ẢNH SỬ DỤNG THUẬT TOÁN KMEANS++

**euclide = 1e-2**



**euclide = 1e-4**



**K= 2**

**euclide = 1e-6**



**euclide = 1e-10**



# DEMO

repo:



```

    < backend
      > __pycache__
      < uploads
        |__ ef8ff036f4354f2e9d279e653a9fde35.png
      < utils
        > __pycache__
        __ ImageProcessing.py
        __ ImageProcessing2.py
        __ main.py
  
```

Backend: Python + FastAPI

```

    < frontend
      > node_modules
      > public
      < src
        > api
        < assets
          __ loading.gif
        < component
          __ Image.jsx
        < pages
          __ BetaHome.jsx
          __ Home.jsx
        # App.css
        __ App.jsx
        # index.css
        __ main.jsx
        __ .env
        __ .gitignore
        __ eslint.config.js
        __ index.html
        { } package-lock.json
        { } package.json
        __ README.md
        __ vite.config.js
        __ .gitignore
  
```

Frontend: React (Vite) + axios + MUI

# THANK YOU FOR YOUR ATTENTION

Thank you for joining us on this exploration of technology and its transformative power. We hope this presentation has inspired new ideas and perspectives about how technology shapes our world.

