Phương pháp Hu's moments

Trí tuệ nhân tạo

Nhóm 3

Ngày 17 tháng 3 năm 2024

Sinh viên thực hiện
106200284 - Hồ Đức Vũ - 20KTMT2
106200240 - Nguyễn Minh Phương - 20KTMT1
106200241 - Huỳnh Vũ Đình Phước - 20KTMT1

Giáo viên hướng dẫn
TS. Hoàng Lê Uyên Thục

Môn học
Trí tuệ nhân tạo

Đề bài
Phương pháp **Hu's moments**

Xuất bản
Đà Nẵng, Ngày 17 tháng 3 năm 2024

Số trang
10

Mục lục

Nội dung báo cáo

1 Giới thiệu phương pháp Hu's moment

Sử dụng phương pháp Hu's moment để mô tả đặc trưng của vật thể. Quá trình tính toán các giá trị làm đặc trưng hình dạng gồm 4 bước sau:

- Bước 1: Tính các mô-men 2 chiều:

$$\mu_{pq} = \sum_x \sum_y x^p y^q \sigma(x, y) \tag{1}$$

  ở đây: (x,y) là tọa độ điểm ảnh, σ(x,y) là hàm ảnh nhị phân, là 1 hoặc là 0 tùy theo điểm ảnh (x,y) thuộc vùng đối tượng hoặc vùng nền tương ứng

- Bước 2: Tính các mô-men trung tâm nhằm làm cho các mô-men 2 chiều ở (1) trở nên bất biến đối với sự dịch chuyển của ảnh nhị phân trong khung hình:

$$m_{pq} = \sum_x \sum_y (x - \overline{x})^p (y - \overline{y})^q s(x, y) \tag{2}$$

  Trong đó $(\overline{x}, \overline{y})$ là trọng tâm của ảnh nhị phân, và được tính bằng công thức: $\overline{x} = \frac{\mu_{10}}{m_{00}}$ và $\overline{y} = \frac{\mu_{01}}{m_{00}}$

- Bước 3: Chuẩn hóa các mô-men trung tâm ở (2) để chúng bất biến đối với sự co giãn của ảnh nhị phân:

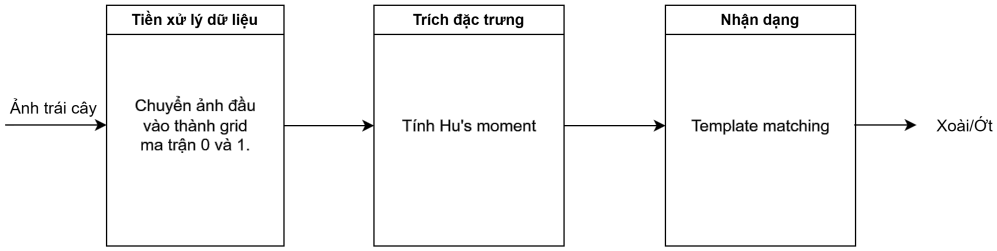
$$M_{pq} = \frac{m_{pq}}{m_{00}^{\frac{p+q}{2}+1}} \tag{3}$$

- Bước 4: Tính 7 mô-men Hu dựa vào các mô-men trung tâm chuẩn hóa ở (3) theo công thức:
  $S_1 = M_{20} + M_{02}$
  $S_2 = (M_{20} - M_{02})(M_{20} + M_{02}) + 4M_{11}M_{11}$
  $S_3 = (M_{30} - 3M_{12})^2 + (M_{30} - 3M_{21})^2$
  $S_4 = (M_{30} + M_{12})^2 + (M_{03} + M_{21})^2$
  $S_5 = (M_{30} - 3M_{12})(M_{30} + M_{12})[(M_{30} + M_{12})^2 - 3(M_{03} + M_{21})^2] + (3M_{21} - M_{03})(M_{03} + M_{21})[3M(M_{30} + M_{12})^2 - (M_{03} + M_{21})^2]$
  $S_6 = (M_{20} - M_{02})[(M_{30} + M_{12})^2 - (M_{03} + M_{21})^2] + 4M_{11}(M_{30} + M_{12})(M_{03} + M_{21})$
  $S_7 = (3M_{21} - M_{03})(M_{30} + M_{12})[(M_{30} + M_{12})^2 - 3(M_{03} + M_{21})^2] + (M_{30} - 3M_{21})(M_{21} + M_{02})[3(M_{30} + M_{12})^2 - (M_{03} + M_{21})^2]$

2 Quá trình triển khai phương pháp Hu's moment của nhóm

Nhóm thực hiện phương pháp Hu's moment nhận dạng trái cây trong ảnh tỉnh. Ảnh đầu vào của nhóm được lựa chọn từ nội dung bài tập trước đó là ảnh quả xoài và trái ớt.

Trước hết nhóm thực hiện sàng lọc dữ liệu với ba bộ ảnh bao gồm một ảnh dùng để train và hai ảnh còn lại dùng để test, tất cả các ảnh đều có nền đen, ảnh train là ảnh chứa một quả xoài, hai ảnh test còn lại mỗi ảnh chứa một quả xoài và một quả ớt tương ứng. Sau đó sử dụng hệ thống như hình 2.1 để thực hiện nhận dạng/phân loại các ảnh test đầu vào.



Hình 2.1: Sơ đồ khối tổng quát hệ thống

Hình 2.1 mô tả tổng quan về hệ thống mà nhóm triển khai, với đầu vào gồm các ảnh chứa trái cây có background đen. Đầu tiên ảnh sẽ được tiền xử lý để chuyển về thành dạng ma trận 0 và 1, sau đó nhóm sẽ sử dụng phương pháp Hu's moment để trích xuất đặc trưng cho ma trận đã thu được trước đó, cuối cùng dùng phương pháp Template maching để thực hiện nhận dạng đối tượng có trong ảnh.

2.1 Tiền xử lý dữ liệu

Bước này nhằm chuyển đổi dữ liệu đầu vào thành grid ma trận 0 và 1.

Với ảnh đầu vào gồm ba ảnh bao gồm một ảnh train và 2 ảnh test như mô tả tại hình 2.2.

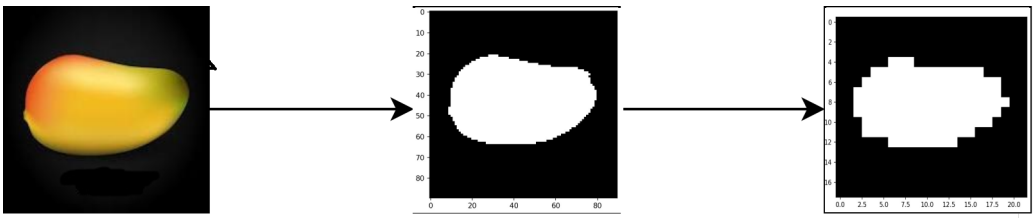
(a) Ảnh train (b) Ảnh test 1 (c) Ảnh test 2

Hình 2.2: Ảnh đầu vào

Để thực hiện bước tiền xử lý dữ liệu, nhóm đã chia ra các bước thực hiện như sau:

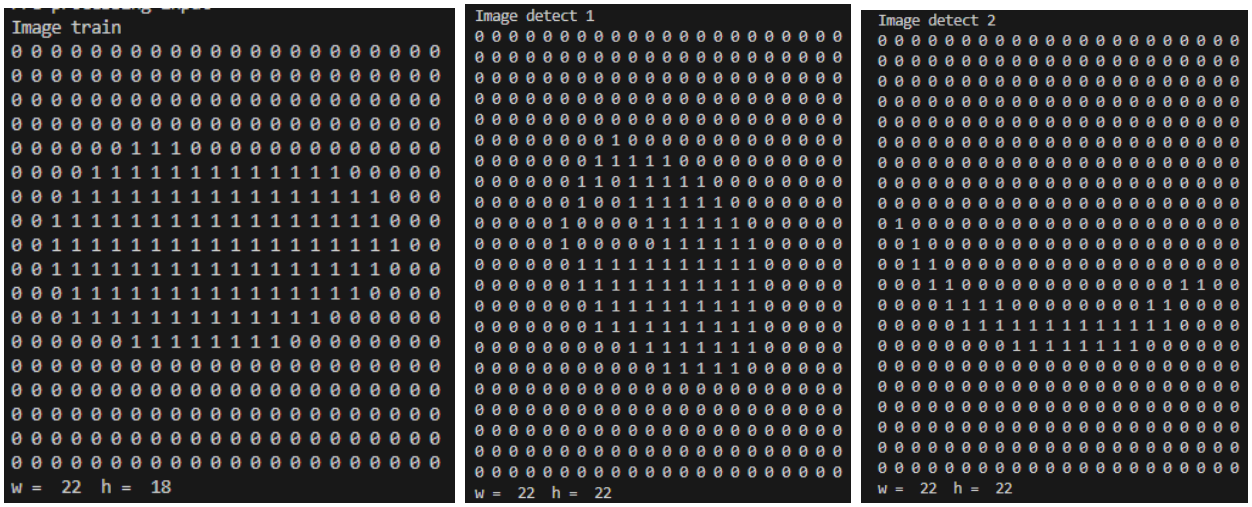
Bước 1: Hình ảnh đầu vào sẽ được thay đổi kích thước về 90x90, sau đó sẽ chuyển sang ảnh nhị phân, như mô tả tại hình 2.3. Đặc điểm của bước này là việc xác định threshold sẽ

ảnh hưởng lớn đến giá trị của các đặc trưng Hu's moment, sau nhiều lần thử nghiệm thì nhóm đã rút ra được ngưỡng ảnh hưởng tốt cho các đặc trưng Hu's moment là 80.



Hình 2.3: Chuyển ảnh đầu vào về dạng ảnh nhị phân

Bước 2: Sau khi có được ảnh nhị phân, nhóm thực hiện bước tiếp theo là chuyển ảnh nhịn phân này về dạng ma trận 0 và 1 (nhóm gọi ma trận này là grid). Sau khi thực hiện bước này thì ta sẽ có 3 ma trận grid tương ứng với 3 ảnh đầu vào. Kết quả của bước tiến xử lý ảnh đầu vào được thể hiện tại hình 2.4. Sau khi có được grid của các ảnh đầu vào, bước tiếp theo nhóm thực hiện trích xuất đặc trưng Hu's moment trên các grid này.



(a) Grid ảnh train (b) Grid ảnh test 1 (c) Grid ảnh test 2

Hình 2.4: Kết quả tiền xử lý dữ liệu

2.2 Trích đặc trưng sử dụng Hu's moment

Trích đặc trưng chính là chuyển đổi đối tượng trích được thành một vec-tơ đặc trưng đa chiều sao cho vec-tơ này chứa đựng các đặc điểm hữu hiệu và riêng biệt, giúp phân biệt đối tượng này với đối tượng khác mà không cần phải dùng toàn bộ khung hình. Bằng cách tính các giá trị Hu's moments như đã được trình bày tại phần 1, ta có thể thu được một vector chứa 7 đặc trưng của đối tượng.

Để thực hiện tính Hu's moment, nhóm triển khai theo các bước như sau:

Bước 1: Nhóm tiến hành tính m_{00} của các grid theo công thức (2), sau khi có được giá trị

m_{00}, tiếp đến tính các giá trị centroid của ảnh gồm $\overline{x}$ và $\overline{y}$.

Bước 2: Trong bước này, nhóm tính toán các giá trị của M_{pq} với p,q=0,1,2,3 như đã đề cập tại phần 1.

Bước 3: Tính toán 7 đặc trưng Hu's moment từ S1 đến S7, kết quả của 7 đặc trưng cho 3 bộ ảnh đầu vào được thể hiện tại bảng 1. Sau khi có được 7 đặc trưng của 3 grid, bước tiếp theo nhóm tiến hành sử dụng phương pháp Template maching để tiến hành nhận dạng đối tượng trong 2 ảnh test đầu vào.

Mô-men	Ảnh train	Ảnh test 1	Ảnh test 2
S_1	0.20441840277777773	0.19306380847952181	0.806997084548105
S_2	0.02575904227249709	0.003064752439784553	0.5930365781604604
S_3	0.00023058998706167655	0.00017602408592489686	0.17706037385406362
S_4	6.565033579966799e-05	0.0001826133043575163	0.030141282633794408
S_5	3.2066518159889194e-09	-1.716501040933791e-08	-0.0018653888790535096
S_6	4.497358340891996e-06	6.195620712386146e-06	-0.020420837124001407
S_7	2.728105714117218e-08	5.77676024148331e-07	8.366576044963944e-05

Bảng 1: Giá trị đặc trưng Hu's moment

2.3 Sử dụng hương pháp Template maching để nhận dạng/phân loại đối tượng trong ảnh

Để thực hiện nhận dạng nhóm sử dụng phương pháp template matching. Phương pháp thực hiện bằng cách so sánh các vector đặc trưng của mẫu huấn luyện với các vector của các mẫu kiểm tra.

Với phương pháp Template maching này có 2 cách để thực hiện, gồm:

- Manhattan distance: $\sum_{i=1}^{n} |x_i - y_i|$
- Euclidean distance: $\sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}$

Trong bài toán này nhóm đã sử dụng phương pháp Template maching thứ nhất là Manhattan distance để tiến hành nhận diện đối tượng trong hai ảnh test.

Sau khi có được hai giá trị từ việc thực hiện phương pháp ở trên để so sánh 7 đặc trưng của ảnh train với 2 ảnh test, nhóm tiến hành so sánh hai giá trị này với nhau, nếu giá trị nào nhỏ hơn thì ảnh tương ứng với giá trị đó là ảnh có đối tượng tưng ứng với ảnh train (quả xoài), ảnh còn lại là quả ớt.

Kết quả của nhóm tại bước này có giá trị như sau:

- norm distance for image test 1: 0.03422268202966381
- norm distance for image test 2: 1.3991359988707275

Ta có thể thấy, giá trị norm distance cho ảnh test thứ nhất nhỏ hơn so với ảnh test thứ hai, vì vậy có thể kết luận là ảnh thứ nhất chứa quả xoài và ảnh thứ hai chứa quả ớt.

3 Kết luận

Nhóm đã thực hiện nhận diện ảnh tỉnh với 2 loại trái cây là xoài và ớt bằng phương pháp Hu's moments kết hợp với template matching.

Với phương pháp Hu's moment, ta có thể nhận dạng đối tượng đơn trong ảnh có nền trơn. Ưu điểm của phương pháp này là ít phụ thuộc vào vị trí cũng như kích thước của đối tượng trong ảnh, tuy nhiên các đặc trưng trích ra từ việc sử dụng phương pháp này dễ bị tác động bởi các yếu tố khác, như trong bài toán nhóm thực hiện, việc lấy ngưỡng nhị phân cũng như điều chỉnh kích thước của ảnh đầu vào ảnh hưởng lớn đến giá trị của các đặc trưng, từ đó ảnh hưởng đến độ chính xác và độ đáng tin cậy của hệ thống.

Đối với các đối tượng có tính tương đồng cao như quả xoài và quả cam, việc sử dụng phương pháp Hu's moment để phân loại các đối tượng có trong ảnh là một phương pháp có độ tin cậy thấp và không chính xác.

Các hệ số đặc trưng Hu's moments trong bài toán nhóm thực hiện cho ra kết quả rất nhỏ gần như bằng 0 gây khó khăn cho việc nhận diện một số ảnh, ta có thể khắc phục bằng cách chuyển các đặc trưng đó sang hệ cơ số Logarit. Hệ thống tốn ít tài nguyên và thời gian tính toán, có thể sử dụng hệ thống để thực hiện các ứng dụng có ít tài nguyên.

Từ những nhận định trên, ta có thể thấy rằng phương pháp Hu's moment có nhiều ưu và cũng như nhược điểm, phương pháp này dễ dàng thực hiện và hiệu quả đối với ảnh có đối tượng đơn trên nền trơn, khi dùng để phân loại với các đối tượng có tính tương đồng thấp thì phương pháp cho ra các kết quả đáng tin cậy, nhưng ngược lại, phương pháp Hu's moment không thể sử dụng cho ảnh có nhiều đối tượng và khi cần phân biệt các đối tượng đơn có tính tương đồng cao thì việc sử dụng phương pháp này trở nên kém hiệu quả.

4 Code Hu's moment

Link repo GitHub chứa ảnh và file code: Click here

```
import numpy as np
import cv2
from PIL import Image
def Image_to_grid(image):
    # Open image, convert to black and white mode
    image = Image.fromarray(image).convert('1')
    w, h = image.size
    # Temporary NumPy array of type bool to work on
    temp = np.array(image, dtype=bool)
```

```
10      # Detect changes between neighboring pixels
11      diff_y = np.abs(np.diff(temp, axis=0).astype(bool))  #
           Convert diff_y to bool
12      diff_x = np.abs(np.diff(temp, axis=1).astype(bool))  #
           Convert diff_x to bool
13      # Create a union image of detected changes
14      temp = np.zeros_like(temp, dtype=bool)
15      temp[:h-1, :] |= diff_y
16      temp[:, :w-1] |= diff_x
17      # Calculate distances between detected changes
18      diff_y = np.diff(np.nonzero(np.diff(np.sum(temp, axis=0))))
19      diff_x = np.diff(np.nonzero(np.diff(np.sum(temp, axis=1))))
20      # Calculate tile height and width
21      ht = np.median(diff_y[diff_y > 1]) + 2
22      wt = np.median(diff_x[diff_x > 1]) + 2
23      # Resize image w.r.t. tile height and width
24      if (ht > 0 and wt > 0):  # Ensure that ht and wt are greater
           than 0
25          array = np.array(image.resize((int(w/wt), int(h/ht)))).
               astype(int)
26      return array, int(w/wt), int(h/ht)
27
28  def convert_to_binary(img_grayscale, thresh=80):
29      thresh, img_binary = cv2.threshold(img_grayscale, thresh,
           maxval=255, type=cv2.THRESH_BINARY)
30      return img_binary
31
32  def Humoment(grid, h, w):
33      # m00
34      m00 = np.sum(grid)
35      xs =0
36      ys = 0
37      centroid = np.array([0.0, 0.0])
38      for y in range(0, int(h)):
39          for x in range(0, int(w)):
40              if grid[y][x] == 1:
41                  ys += y + 1
42                  xs += x + 1
43                  centroid += np.array([y + 1, x + 1])
44      print('ys = ', ys, ' xs = ', xs)
45      centroid /= float(m00)
46      print("m00 =", m00)
47      print("centroidx =", centroid[1])
48      print("centroidy =", centroid[0])
49      # Calculate central moments
50      p0, p1, p2, p3 = 0, 1, 2, 3
```

```
    q0, q1, q2, q3 = 0, 1, 2, 3
    mqp1, mqp2, mqp3, mqp4, mqp5, mqp6, mqp7 = 0, 0, 0, 0, 0, 0,
       0

    for y in range(0, int(h)):
        for x in range(0, int(w)):
            if grid[y][x] == 1:
                mqp1 += (float(x + 1) - centroid[1]) ** p2 * (
                   float(y + 1) - centroid[0]) ** q0
                # M02
                mqp2 += (float(x + 1) - centroid[1]) ** p0 * (
                   float(y + 1) - centroid[0]) ** q2
                # M11
                mqp3 += (float(x + 1) - centroid[1]) ** p1 * (
                   float(y + 1) - centroid[0]) ** q1
                # M30
                mqp4 += (float(x + 1) - centroid[1]) ** p3 * (
                   float(y + 1) - centroid[0]) ** q0
                # M12
                mqp5 += (float(x + 1) - centroid[1]) ** p1 * (
                   float(y + 1) - centroid[0]) ** q2
                # M03
                mqp6 += (float(x + 1) - centroid[1]) ** p0 * (
                   float(y + 1) - centroid[0]) ** q3
                # M21
                mqp7 += (float(x + 1) - centroid[1]) ** p2 * (
                   float(y + 1) - centroid[0]) ** q1
    M20 = mqp1 / (m00 ** ((p2 + q0) / 2 + 1))
    M02 = mqp2 / (m00 ** ((p0 + q2) / 2 + 1))
    M11 = mqp3 / (m00 ** ((p1 + q1) / 2 + 1))
    M30 = mqp4 / (m00 ** ((p3 + q0) / 2 + 1))
    M03 = mqp5 / (m00 ** ((p0 + q3) / 2 + 1))
    M12 = mqp6 / (m00 ** ((p1 + q2) / 2 + 1))
    M21 = mqp7 / (m00 ** ((p2 + q1) / 2 + 1))
    S1 = M20 + M02
    S2 = (M20 + M02) * (M20 - M02) + 4 * M11 ** 2
    S3 = (M30 - 3 * M12) ** 2 + (M30 - 3 * M21) ** 2
    S4 = (M30 + M12) ** 2 + (M03 + M21) ** 2
    S5 = (M30 - 3 * M12) * (M30 + M12) * ((M30 + M12) ** 2 - 3 *
       (M03 + M21) ** 2) + (3 * M21 - M03) * (M03 + M21) * (3 * (
       M30 + M12) ** 2 - (M03 + M21) ** 2)
    S6 = (M20 - M02) * ((M30 + M12) ** 2 - (M03 + M21) ** 2) + 4
       * M11 * (M30 + M12) * (M03 + M21)
    S7 = (3 * M21 - M03) * (M30 + M12) * ((M30 + M12) ** 2 - 3 *
       (M03 + M21) ** 2) + (M30 - 3 * M12) * (M21 + M02) * (3 * (
       M30 + M12) ** 2 - (M03 + M12) ** 2)
```

```
84
85      print("S1 =", S1)
86      print("S2 =", S2)
87      print("S3 =", S3)
88      print("S4 =", S4)
89      print("S5 =", S5)
90      print("S6 =", S6)
91      print("S7 =", S7)
92      return S1, S2, S3, S4, S5, S6, S7
93
94  def Template_matching(S1, S2):
95      norm_distance = float(0)
96      for i in range(0, 7):
97          # Use 1-norm-distance
98          norm_distance += float(abs(S1[i] - S2[i]))
99      return norm_distance
100
101 # Pre-processing image input
102 print("\nPre-processing input")
103 # Image train
104 path_image = 'images/mangowithblackground.jpg'
105 image = cv2.imread(path_image, cv2.IMREAD_GRAYSCALE)
106 image_resize = cv2.resize(image,(90,90))
107 binnaryImage = convert_to_binary(image_resize)
108 grid, w, h = Image_to_grid(binnaryImage)
109 # Image detect 1
110 path_image_detect_1 = 'images/mangotest4.jpg'
111 image_detect_1 = cv2.imread(path_image_detect_1, cv2.
      IMREAD_GRAYSCALE)
112 image_resize_detect_1 = cv2.resize(image_detect_1,(90,90))
113 binnaryImage_detect_1 = convert_to_binary(image_resize_detect_1)
114 grid_detect_1, w_detect_1, h_detect_1 = Image_to_grid(
      binnaryImage_detect_1)
115 # Image detect 2
116 path_image_detect_2 = 'images/chillitest2.jpg'
117 image_detect_2 = cv2.imread(path_image_detect_2, cv2.
      IMREAD_GRAYSCALE)
118 image_resize_detect_2 = cv2.resize(image_detect_2,(90,90))
119 binnaryImage_detect_2 = convert_to_binary(image_resize_detect_2)
120 grid_detect_2, w_detect_2, h_detect_2 = Image_to_grid(
      binnaryImage_detect_2)
121
122 # Result pre-processing input
123 print("Image train")
124 for y in range(0, int(h)):
125     for x in range(0, int(w)):
```

```
126             print(grid[y][x], end=" ")
127         print("")
128 print('w = ',w, ' h = ', h)
129 print("Image detect 1")
130 for y in range(0, int(h_detect_1)):
131         for x in range(0, int(w_detect_1)):
132             print(grid_detect_1[y][x], end=" ")
133         print("")
134 print('w = ',w_detect_1, ' h = ', h_detect_1)
135 print("Image detect 2")
136 for y in range(0, int(h_detect_2)):
137         for x in range(0, int(w_detect_2)):
138             print(grid_detect_2[y][x], end=" ")
139         print("")
140 print('w = ',w_detect_2, ' h = ', h_detect_2)
141 # Hu'momnet
142 print("\nHu\'moment:")
143 print("S_train")
144 S_train = Humoment(grid, h, w)
145 print("\nS_detect_1")
146 S_detect_1 = Humoment(grid_detect_1, h_detect_1, w_detect_1)
147 print("\nS_detect_2")
148 S_detect_2 = Humoment(grid_detect_2, h_detect_2, w_detect_2)
149 # Use Template matching to implementation of classification
150 print("\nManhattan distance:")
151 norm_distance_1 = Template_matching(S_train, S_detect_1)
152 norm_distance_2 = Template_matching(S_train, S_detect_2)
153 # Result
154 print('norm distance for image detect 1: ',norm_distance_1)
155 print('norm distance for image detect 2: ',norm_distance_2)
156 # Classification
157 print("\nClassification:")
158 if norm_distance_2 > norm_distance_1 :
159         print("First image detect has a mango and Second image detect
              has a chilli")
160 else:
161         print("First image detect has a chilli and Second image
             detect has a mango")
162 print("")
```