**HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**SCHOL OF INFORMATION TECHNOLOGY AND COMMUNICATION**

**-----□□&□□-----**



**Báo cáo môn: Xử lý ảnh**

**Giáo viên hướng dẫn:**  **TS. Nguyễn Thị Oanh**

**Nhóm thực hiện :** Nhóm 6

|  |  |
| --- | --- |
| **Viên Quốc Anh** | **20176000** |
| **Võ Trí Anh** | **20172176** |
| **Lương Ngọc Đức** | **20175853** |
| **Vũ Hải Long** | **20176041** |
| **Nguyễn Minh Quân** | **20176050** |

***Hà Nôi, 1/ 2022***

***Bài 2: Xác định góc tạo bởi 2 đồ vật trong ảnh***

* *Tập dữ liệu: Cung cấp từ Teams*
* *File code: test5.py*

1. Yêu cầu đề bài:

Cho tập hợp ảnh, mỗi ảnh gồm 2 vật thể là “điều khiển” và “điện thoại”, tính góc tạo bởi hai vật thể trong ảnh.

* + Định nghĩa: Góc tạo bởi hai vật thể là góc nhỏ hơn tạo ra bởi trục dọc của hai vật.

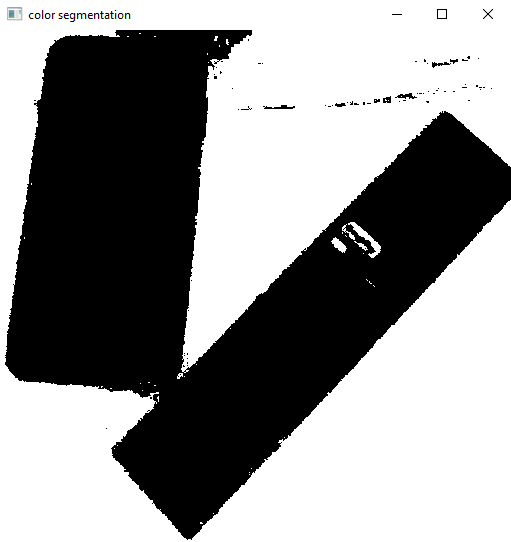
A picture containing electronics, indoor, sitting, floor

Description automatically generated

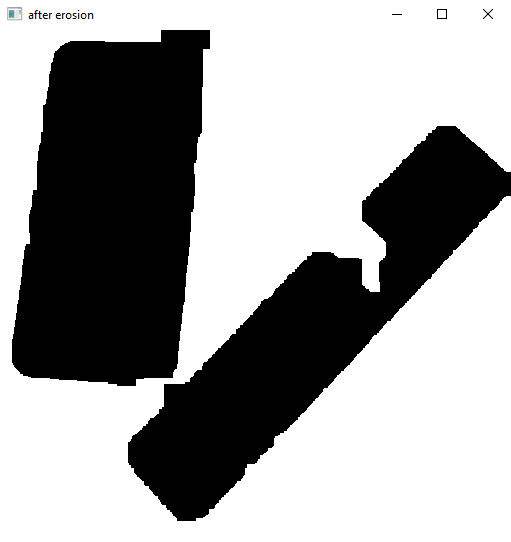
* + Lưu ý: Ảnh sử dụng có thể có nhiễu (muối tiêu, hình sin...)

1. Các bước giải quyết:

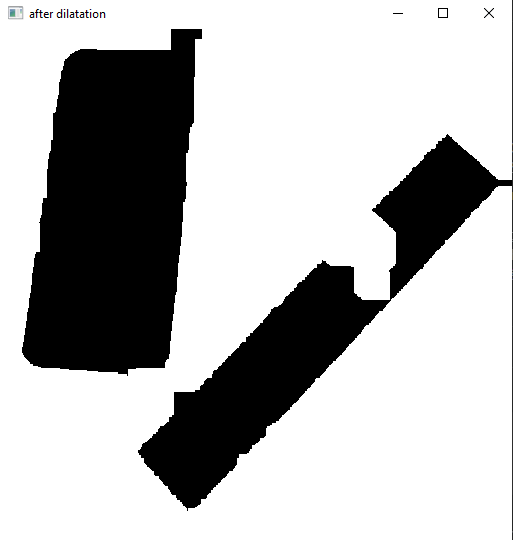
* **Bước 1:** Đọc ảnh và chuyển ảnh về miền màu HSV. Nếu ảnh có nhiễu muối tiêu (file *img2.png\_noise.png*) thì sử dụng hàm *cv2.bilateralFilter* để lọc nhiễu.
* **Bước 2:** Sử dụng phương pháp lấy ngưỡng màu để tạo mask che background và chuyển về ảnh nhị phân. *Ngưỡng màu sử dụng từ [0,0,0] tới [180,255,255] (áp dụng cho mặt bàn)*:



* **Bước 3:** Áp dụng Morphological Transformations để loại bỏ các phần nhiễu còn lại.
  + Sử dụng Erosion:



* + Sử dụng Dilation:



* **Bước 4:** Tạo ra một mask bao quanh ảnh gốc màu trắng ==> Phân tách vật thể với biên ảnh dễ hơn.
* **Bước 5:** Sử dụng hàm *cv2.findContours* của thư viện OpenCV để xác định biên của các vật thể trong ảnh ==> các mảng con lưu giữ giá trị vị trí các điểm biên.
* **Bước 6:** Sử dụng hàm *sorted* của Python để sắp xếp các mảng từ bước 4 theo thứ tự giảm dần. Hai mảng con có giá trị lớn nhất (biên rộng nhất) là hai vật thể cần tìm ==> cnt[1] và cnt[2].
* **Bước 7:** Sử dụng hàm *cv2.PCACompute2* với đầu vào là mảng cnt[1] và cnt[2] (mảng cnt[0] là toàn bộ hình ảnh) để tìm vị trí tâm (mean) và góc chếch của vật thể (tính bằng eigenvalues, đổi từ radian sang degree).
* **Bước 8:** Tính góc từ kết quả từ bước 7, in kết quả ra màn hình trên ảnh gốc. Vẽ thêm mũi tên để tăng tính trực quan.



1. Kết quả thử nghiệm:

Chương trình đã nhận diện được góc của các vật thể trong ảnh.

***Bài 3: Phân loại ảnh sử dụng phương pháp láng giềng gần nhất***

1. Yêu cầu đề bài:
   * Cho tập ảnh chứa các mẫu bề mặt đồ vật. Các mẫu được phân chia phần các loại và được chia thành dữ liệu học và dữ liệu kiểm thử. Xây dựng chương trình để nhận dạng mẫu
   * Sử dụng tập dữ liệu: <https://github.com/abin24/Textures-Dataset>
     + Tập dữ liệu gồm 64 class, được dán nhãn từ 0 - 63
     + Folder train là tập dữ liệu học
     + Folder valid là tập dữ liệu kiểm thử

Text

Description automatically generated

1. Các bước thực hiện:

* Bước 1: Đọc ảnh và dán nhãn cho tập dữ liệu train và test
  + Đọc ảnh và dán nhãn dữ liệu tập train

Text

Description automatically generated

* + Đọc ảnh và dán nhãn dữ liệu cho tập test

Text

Description automatically generated

* Bước 2: Trích xuất đặc trưng cho từng ảnh
  + Cách 1: Sử dụng các ma trận đồng xuất hiện với các thuộc tính Haralick

Text

Description automatically generated

*Haralick Texture Features được tính toán bằng cách trích xuất các đặc trưng từ ma trận đồng xuất hiện Gray-Level Co-occurrence Matrix (GLCM). Ma trận này đặc trưng cho kết cấu của bức ảnh bằng cách ghi lại tần suất xuất hiện trong bức ảnh của các cặp điểm ảnh liền kề với các giá trị cụ thể.*

*Áp dụng 9 ma trận, mỗi ma trận gồm các khoảng cách và hướng như sau:*

* *Distance: 1, 3, 5*
* *Direction: 0o, 45o, 90o*

*6 đặc trưng được trích xuất từ ma trận đồng xuất hiện là:*

* *dissimilarity (độ đồng đều)*

A picture containing text, watch, clock

Description automatically generated

* *correlation (tương quan)*

A picture containing text, watch

Description automatically generated

* *energy (năng lượng)*

A picture containing graphical user interface

Description automatically generated

* *contrast (độ tương phản)*

A picture containing object, watch, clock

Description automatically generated

* *homogeneity (tính đồng nhất)*

A picture containing clock, watch

Description automatically generated

* *ASM - Active shape models*

**

* Như vậy tổng cộng chúng ta có 54 đặc trưng từ 9 ma trận đồng xuất hiện
  + Cách 2: Sử dụng local binary patterns

Text

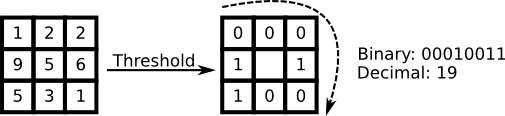
Description automatically generated

*Tương tự như Haralick Texture Features, LBP được dùng để mô tả kết cấu, hoa văn (texture, pattern) của một bức ảnh/đối tượng. Tuy nhiên, thay vì miêu tả kết cấu một cách tổng quát như Haralick Texture, LBP miêu tả theo từng khu vực, bằng cách so sánh giá trị của một pixel với các pixel khác bao quanh nó. Nhờ đó mà LBP mạnh mẽ hơn nhiều so với Haralick Texture Features, tuy nhiên kèm theo đó là yêu cầu về tính toán và kích thước Feature Vector sẽ lớn hơn. LBP đã được áp dụng rất thành công trong bài toán nhận diện khuôn mặt.*

*Bước đầu tiên để xây dựng một LBP descriptor là chuyển đổi input image về grayscale. Với mỗi pixel trong grayscale image, chúng ta chọn ra vùng lân cận (neighborhood) bao quanh pixel trung tâm đó, tất nhiên sẽ là 3×3. Một giá trị LBP sẽ được tính toán cho pixel trung tâm này và lưu vào một ma trận 2D có kích thước giống với input image.*

*Khi so sánh pixel trung tâm và các pixel khác trong vùng lân cận, nếu các pixel này có giá trị lớn hơn hoặc bằng pixel trung tâm thì sẽ được đánh dấu là “1”, ngược lại sẽ được đánh dấu là “0”, xem figure 1.*

*Tính toán giá trị LBP với 3×3 neighborhood:*



*Với 8 pixel xung quanh pixel trung tâm, LBP sẽ có giá trị nằm trong khoảng [0, 255]. Tiếp theo chúng ta tính toán histogram cho ma trận 2D chứa các giá trị LBP của tất cả các pixel có trong input image và tìm ra Feature Vector.*

* Bước 3: Train model KNN sử dụng các đặc trưng và nhãn tập train dự đoán các ảnh trong tập test để tạo ra mô hình

A screenshot of a computer

Description automatically generated with medium confidence

*KNN (K-Nearest Neighbors) là một trong những thuật toán học có giám sát đơn giản nhất được sử dụng nhiều trong khai phá dữ liệu và học máy. Ý tưởng của thuật toán này là nó không học một điều gì từ tập dữ liệu học (nên KNN được xếp vào loại lazy learning), mọi tính toán được thực hiện khi nó cần dự đoán nhãn của dữ liệu mới.   
      Lớp (nhãn) của một đối tượng dữ liệu mới có thể dự đoán từ các lớp (nhãn) của k hàng xóm gần nó nhất.  
Chart, bubble chart

Description automatically generated*

*Các bước trong KNN*

* *Ta có D là tập các điểm dữ liệu đã được gắn nhãn và V là dữ liệu chưa được phân loại.*
* *Đo khoảng cách từ dữ liệu mới V đến tất cả các dữ liệu khác đã được phân loại trong D.*
* *Chọn k (k là tham số mà bạn định nghĩa) khoảng cách nhỏ nhất.*
* *Kiểm tra danh sách các lớp có khoảng cách ngắn nhất và đếm số lượng của mỗi lớp xuất hiện.*
* *Lấy đúng lớp (lớp xuất hiện nhiều lần nhất).*
* *Lớp của dữ liệu mới là lớp mà bạn đã nhận được ở bước 5.*
* Bước 4: Dự đoán nhãn các ảnh trong tập test với k = 3

Text

Description automatically generated

* Bước 5: Sử dụng kết quả đoán được và nhãn tập test tính ra độ chính xác
  + Kết quả nghiệm thu với cách 1:

Text

Description automatically generated

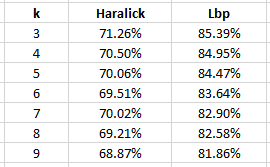
* + Kết quả nghiệm thu với cách 2:

Text

Description automatically generated

1. Kết luận

Bằng cách sử dụng phương pháp KNN và trích xuất đặc trưng ảnh từ ma trận đồng xuất hoặc local binary patterns, ta sẽ nghiệm thu được kết quả phân loại với độ chính xác khá cao (trên 68%), sau đây là thống kê và so sánh độ chính xác của 2 cách trích xuất đặc trưng ảnh với các khoảng cách k tương ứng:



**Hướng dẫn cài đặt và chạy thử nghiệm:**

Bài 2:

* các file ảnh và **test5.py** để cùng một thư mục
* Dùng lệnh python test5.py [tenanh] 1
* Riêng với ảnh có nhiễu muối tiêu dùng lệnh: python test5.py [tenanh] Noise

Bài 3:

* Tải tập dữ liệu trên github, giải nén được **2 folder train và valid**
* Tạo thư mục **data** cùng vị trí với các file code rồi copy 2 folder train và valid vào đây

Graphical user interface, text, application, chat or text message

Description automatically generated

* Trong 2 file **bai3-haralick.ipynb/bai3-haralick.py** hoặc **bai3-lbp.ipynb/ bai3-lbp.py**
  + cài thư viện cần thiết: pip install scikit-image