Trường Đại Học Bách Khoa Hà Nội
Viện Công nghệ thông tin và truyền thông
======*o0o*======

BÁO CÁO

MÔN: Thị giác máy tính

Giáo viên hướng dẫn: TS. Nguyễn Thị Oanh

Sinh viên thực hiện

Vũ Quang Huy	20173178
Nguyễn Thế Đức	20170057
Nguyễn Minh Hiếu	20173115
Nguyễn Kỳ Tùng	20173455
Nguyễn Minh Đăng	20172998

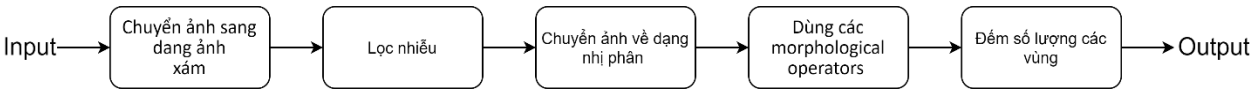
Hà Nội, ngày 24 tháng 06 năm 2021

MỤC LỤC

I. Project 1: Object counting

1. Thuật toán chung

Sau đây là sơ đồ chung của thuật toán đếm số lương vật thể, trong đó đầu ra của bước trước sẽ là đầu vào của bước sau:



Hình 1: Sơ đồ thuật toán đếm số lượng vật thể.

Chi tiết các bước của thuật toán như sau:

- Bước 1: Chuyển ảnh sang dạng ảnh xám:
  Trong bài toán đếm vật thể màu sắc của vật thể không quan trọng nên ta sẽ chuyển ảnh về thang màu xám, giúp thực hiện các bước sau dễ dàng và chính xác hơn.
- Bước 2: Lọc bớt nhiễu muối tiêu bằng cách sử dụng median filter hoặc lọc trong miền tần số (đối với ảnh có nhiễu sinus - Ảnh 3).
- Bước 3: Chuyển ảnh về dạng dạng nhị phân để xác định các vật thể bằng cách sử dụng adaptive threshold:
  - Do ảnh có thể có độ sáng không đồng đều giữa các vùng nên không sử dụng global threshold.
  - Đầu ra là ảnh nền có màu đen, vật màu trắng để xử lý bước tiếp theo, nếu ảnh đầu vào là nền trắng thì tiến hành lấy nghịch đảo pixel.
- Bước 4: Sử dụng các morphological operators (**cv.morphologyEx()**):
  - Loại bỏ các nhiễu bằng opening.
  - Lấp đầy các lỗ trong vật thể bằng closing
- Bước 5: Đếm các vật thể bằng cách đếm các vùng:
  Sau bước 4 mỗi vật thể sẽ được thể hiện bằng một vùng trắng, chỉ cần đếm các vùng trắng sẽ có được số lượng vật thể bằng hàm **cv.findContours()**.

2. Các trường hợp cụ thể

Giải thích ý nghĩa của một số các tham số:

- Median filter kernel size: kích thước của median filter.
- Blocksize: kích thước của vùng lân cận.
- Adaptive method: phương pháp được sử dụng để tìm giá trị threshold, có hai phương pháp chính được OpenCV hỗ trợ đó là:
  - **cv.ADAPTIVE_THRESH_MEAN_C**: giá trị threshold được tính bằng giá trị các pixel lân cận trừ đi một hằng số C
  - **cv.ADAPTIVE_THRESH_GAUSSIAN_C**: giá trị threshold được tính bằng tổng Gausian có trọng số giá trị các pixel lân cận trừ đi một hằng số C
- C: một hằng số được dùng để tính toán giá trị threshold.

2.1 Ảnh 1, 2, 4

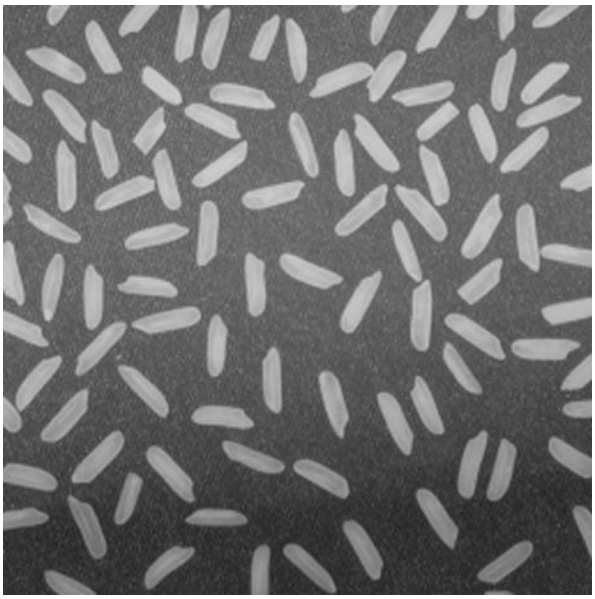
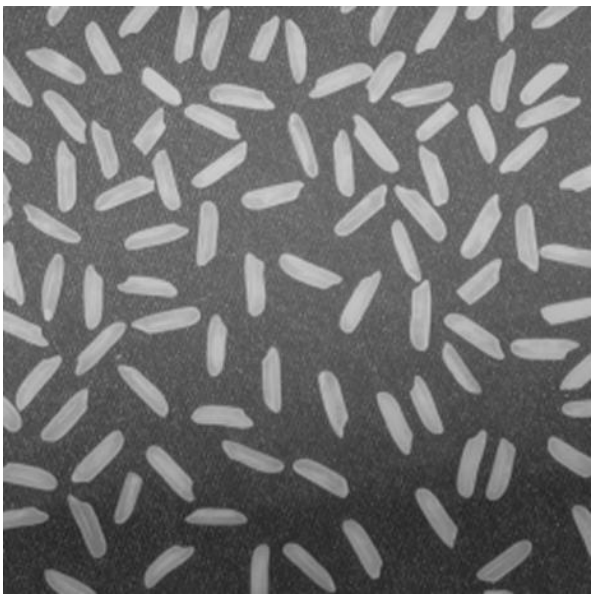
Ba ảnh này được sử dụng chung bộ tham số như sau:

- Median filter: median filter kernel size 3x3.
- Adaptive Threshold:
    - Blocksize: 401
    - Adaptive method: **cv.ADAPTIVE_THRESH_GAUSSIAN_C**
    - C: 0
- Opening: opening kernel size 4x4

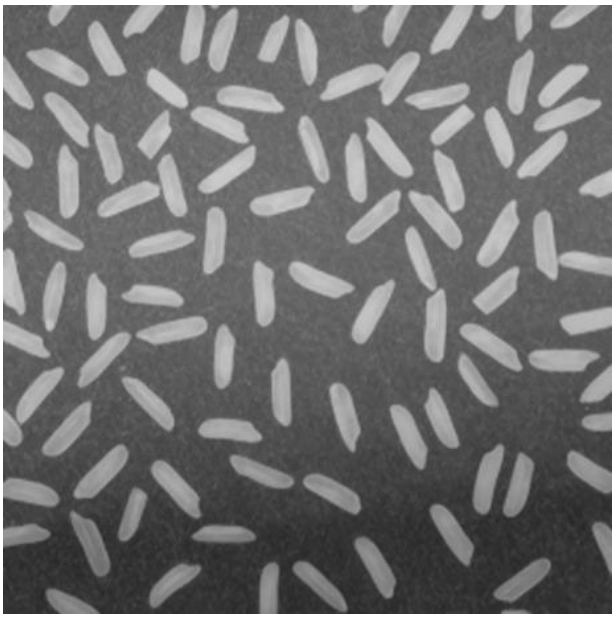
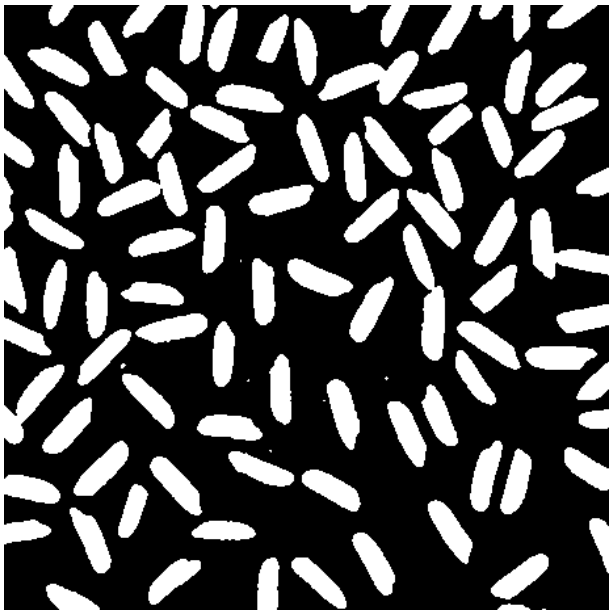
2.1.1 Ảnh 1

Sau đây là các kết quả xử lý qua từng bước của ảnh 1

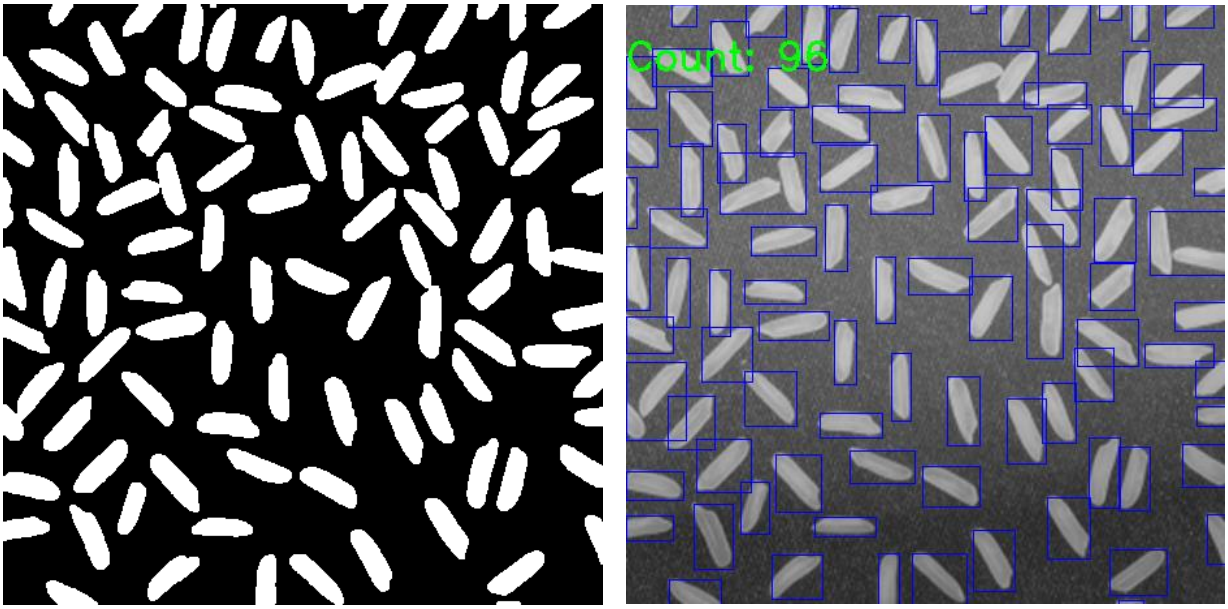
- Ảnh gốc (bên trái) và ảnh sau khi chuyển thành ảnh xám (bên phải):

- Sau khi sử dụng median filter (bên trái) và sau khi sử dụng adaptive threshold (bên phải):

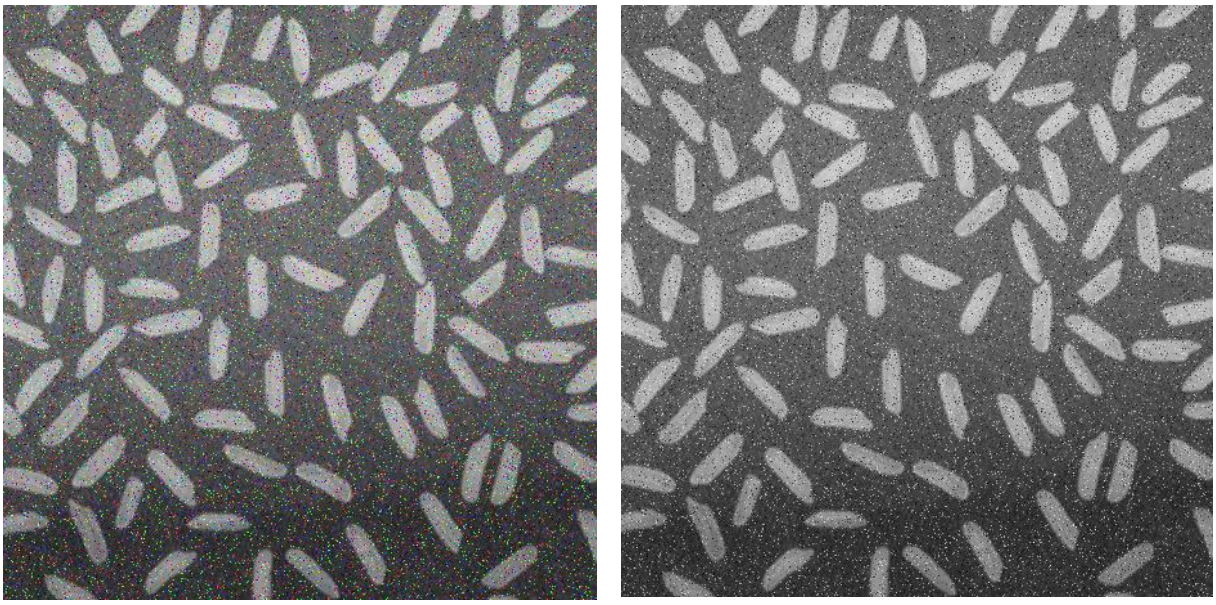
- Sau khi sử dụng opening (bên trái) và kết quả cuối cùng (bên phải):



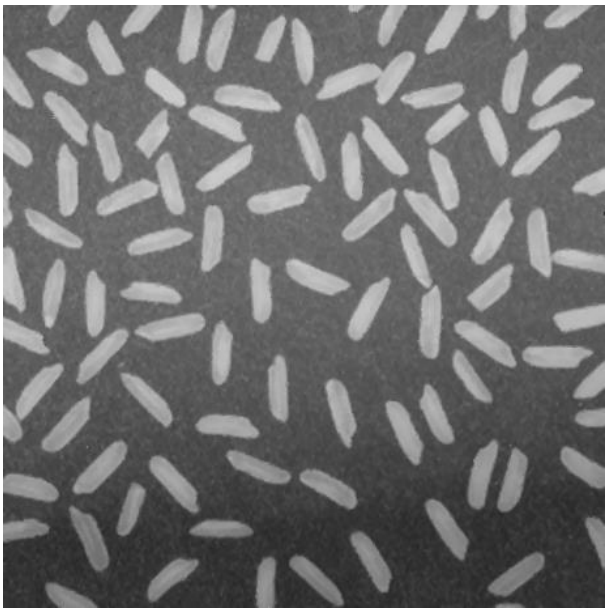
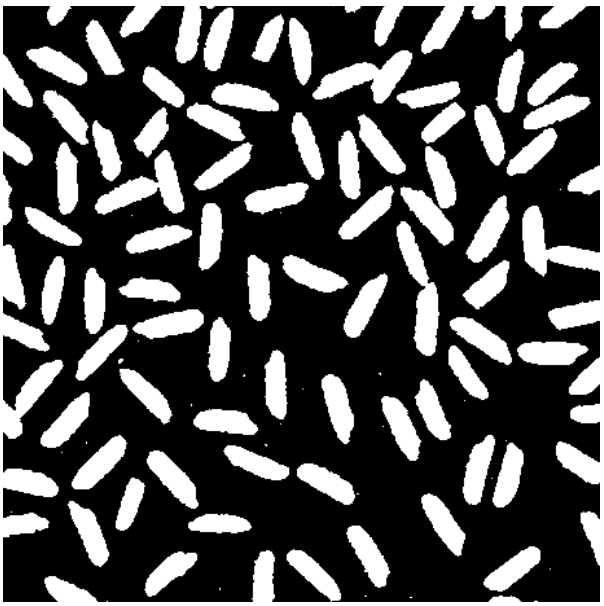
2.1.2 Ảnh 2

Sau đây là các kết quả xử lý qua từng bước của ảnh 2

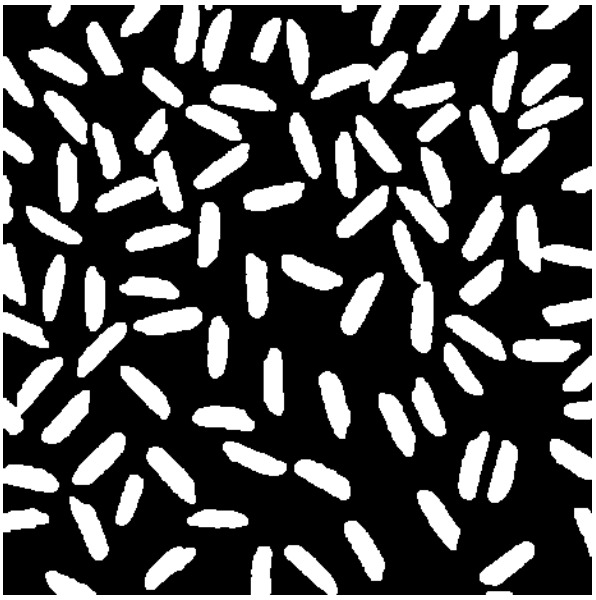
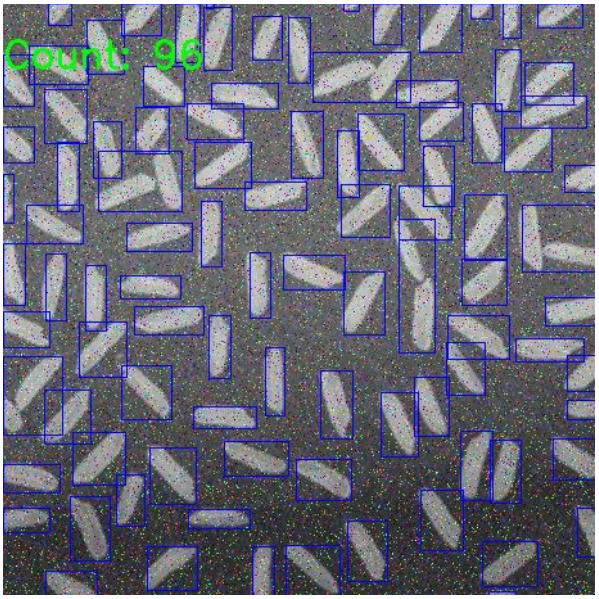
- Ảnh gốc (bên trái) và ảnh sau khi chuyển thành ảnh xám (bên phải):



- Sau khi sử dụng median filter (bên trái) và sau khi sử dụng adaptive threshold (bên phải):

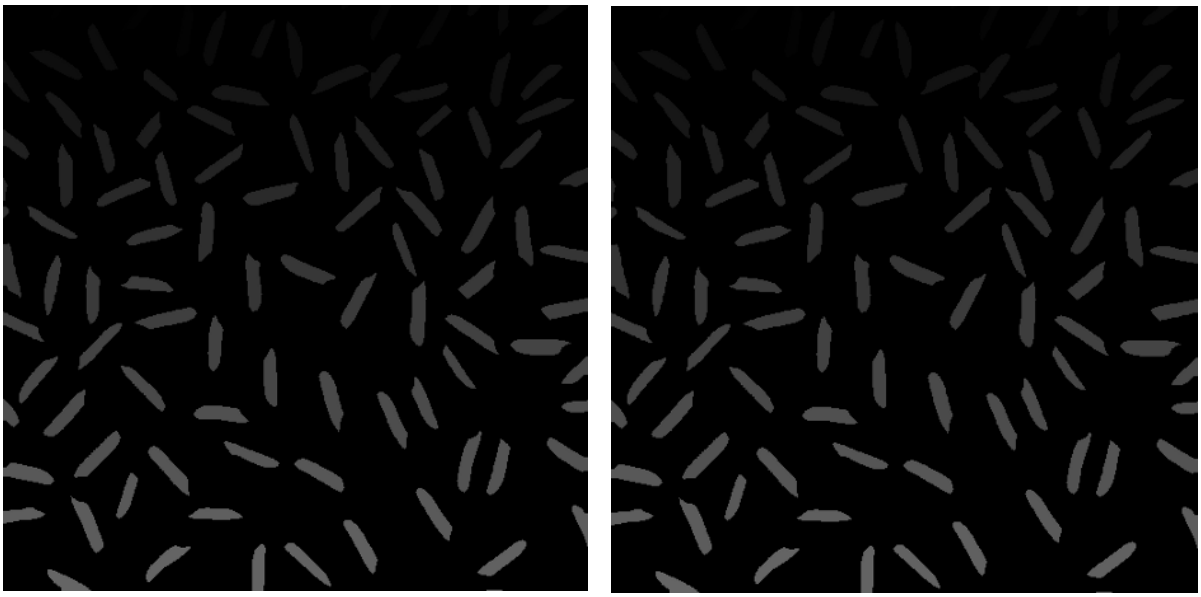
- Sau khi sử dụng opening (bên trái) và kết quả cuối cùng (bên phải):

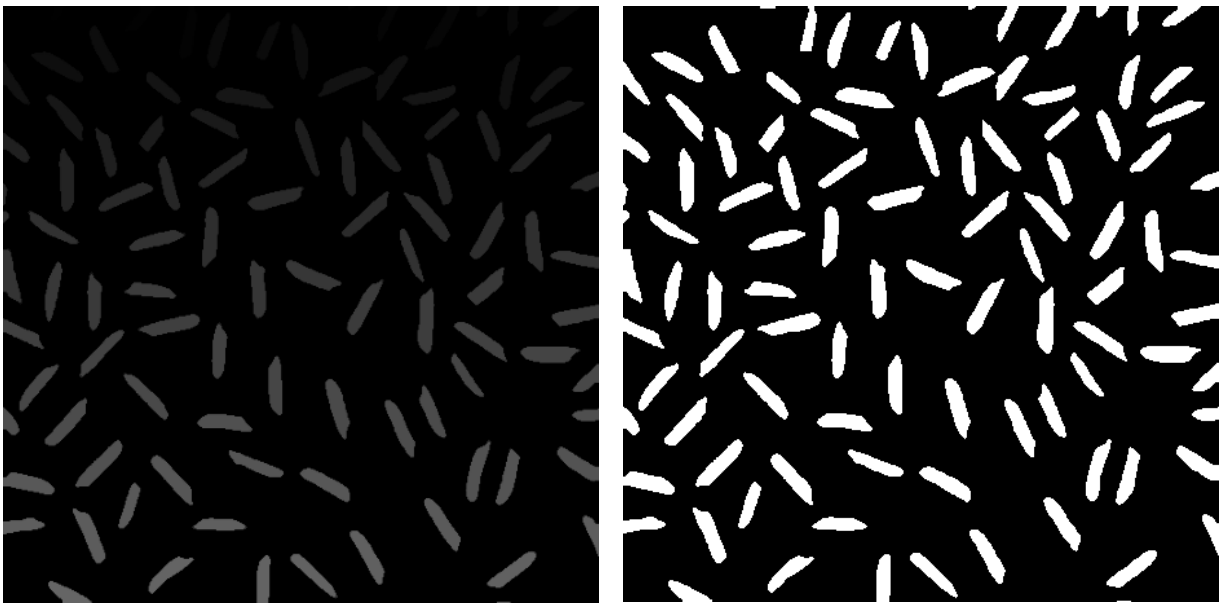
2.1.3 Ảnh 4

Sau đây là các kết quả xử lý qua từng bước của ảnh 4:

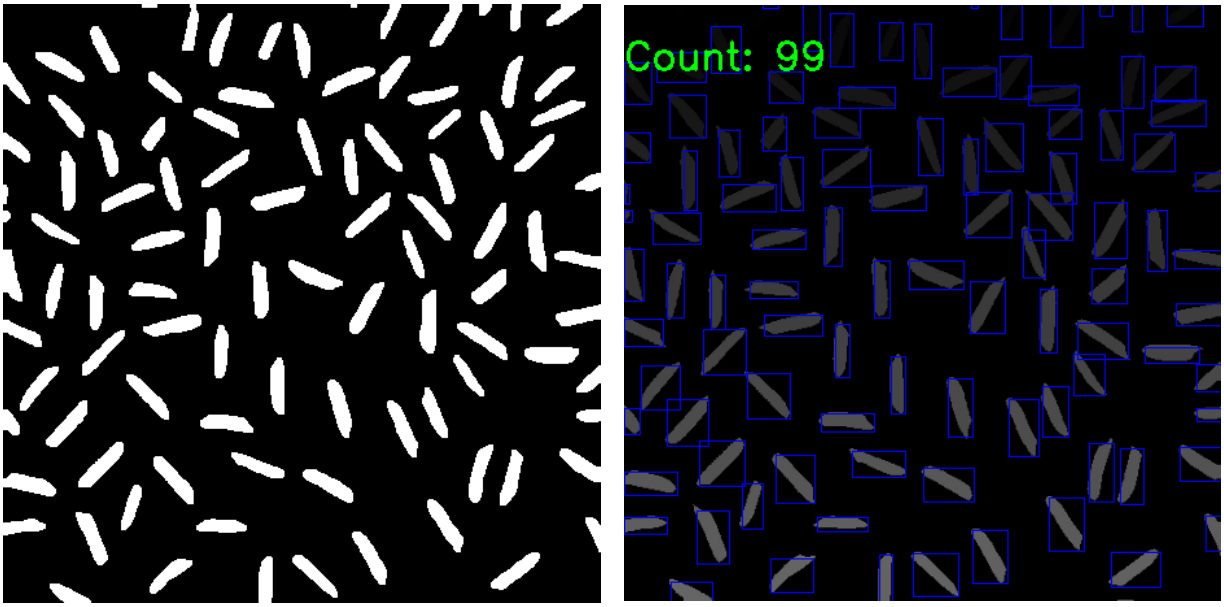
- Ảnh gốc (bên trái) và ảnh sau khi chuyển thành ảnh xám (bên phải):



- Sau khi sử dụng median filter (bên trái) và sau khi sử dụng adaptive threshold (bên phải):



- Sau khi sử dụng opening (bên trái) và kết quả cuối cùng (bên phải):



2.2 Ảnh 3

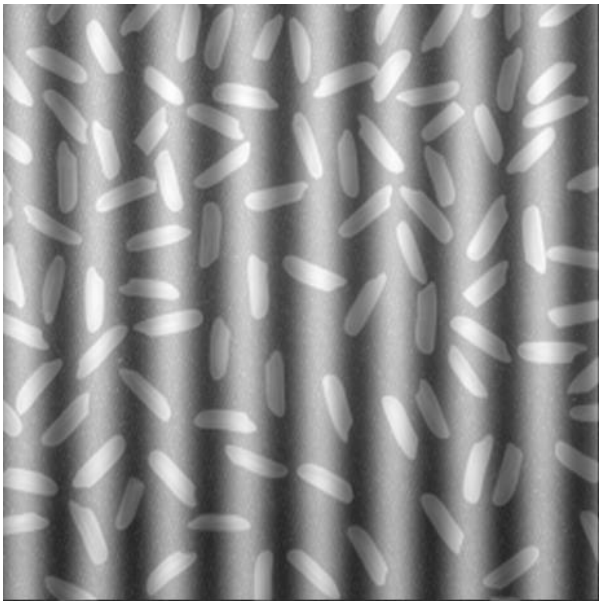
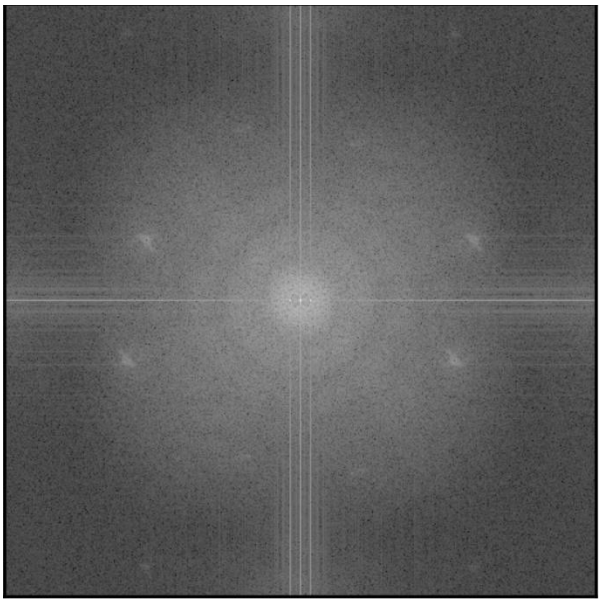
Ảnh này có chứa nhiễu dạng hình sin, cho nên ta cần chuyển nó về dạng tần số để lọc nhiễu. Sau đó chuyển ngược lại ảnh từ miền tần số sang miền không gian rồi tiếp tục các bước theo thuật toán. Hai điểm cần loại bỏ trong miền tần số là các điểm cách tâm ảnh 9pixel theo chiều ngang. Ngoài ra để có kết quả tốt hơn thì có thể loại thêm một số các lân cận của hai điểm cần loại bỏ. Để đảm bảo đúng kiểu dữ liệu cần xử lý, cũng cần chuyển ảnh sau sau khi lọc nhiễu qua miền tần số về dạng uint8 và tiến hành lấy nghịch đảo pixel.

Về chi tiết bộ tham số:

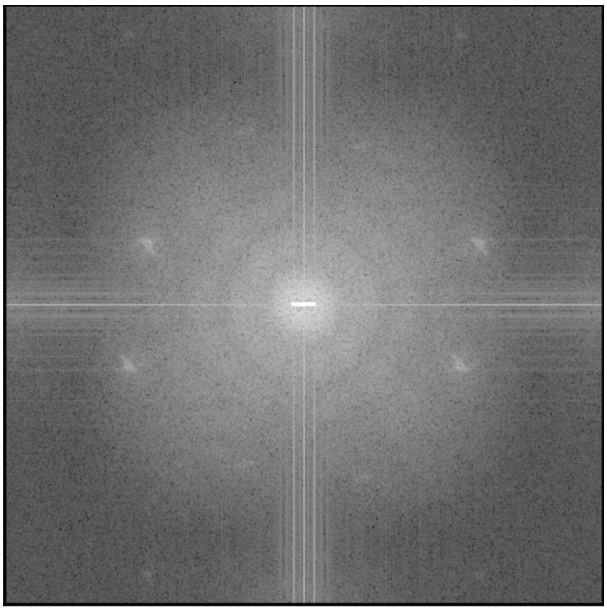
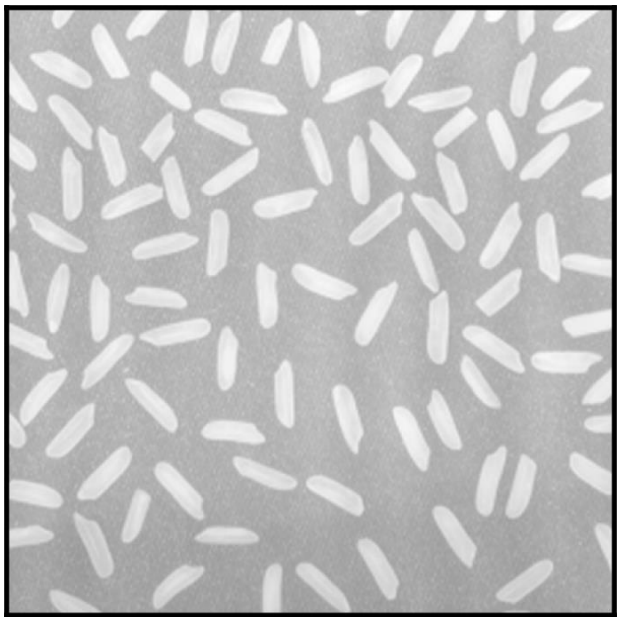
- Adaptive Threshold:
    - Blocksize: 301
    - C: 0
    - Adaptive method: **cv.ADAPTIVE_THRESH_GAUSSIAN_C**
- Opening: closing kernel size 10 x 10

Sau đây là các kết quả xử lý qua từng bước của ảnh 3:

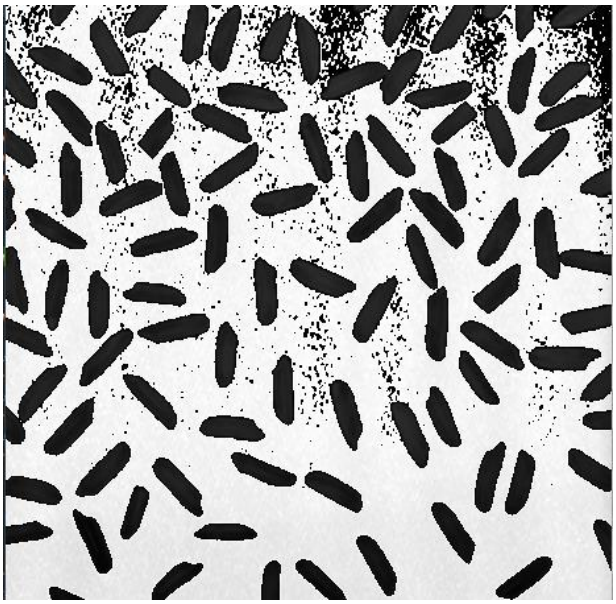
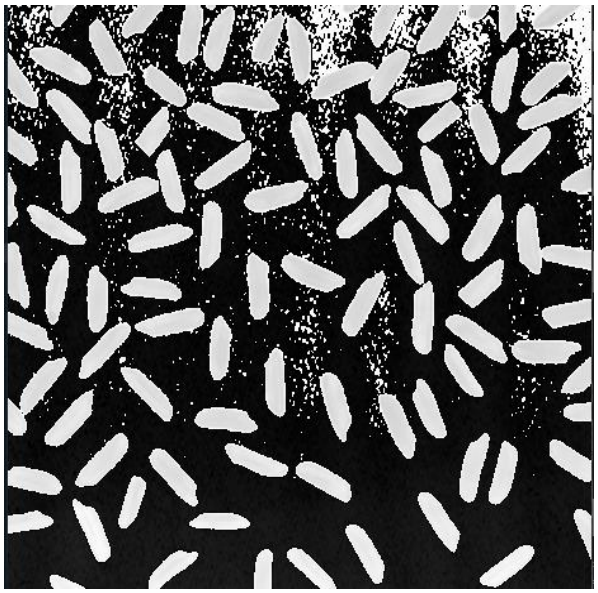
- Ảnh gốc (bên trái) và sau khi chuyển sang miền tần số (bên phải):

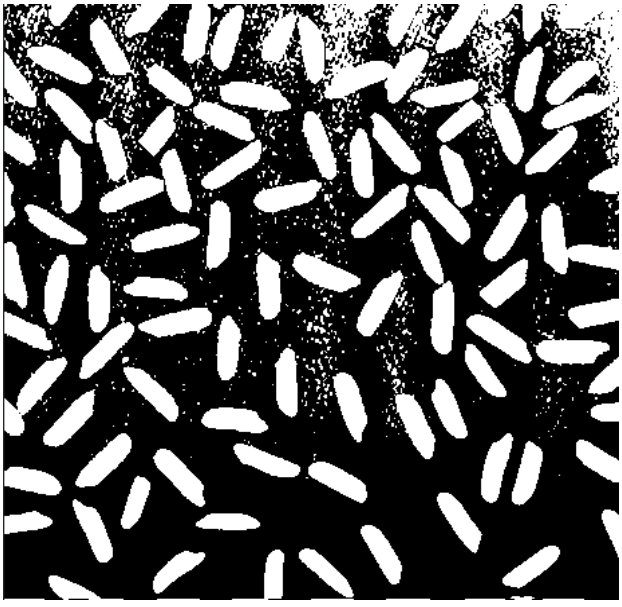
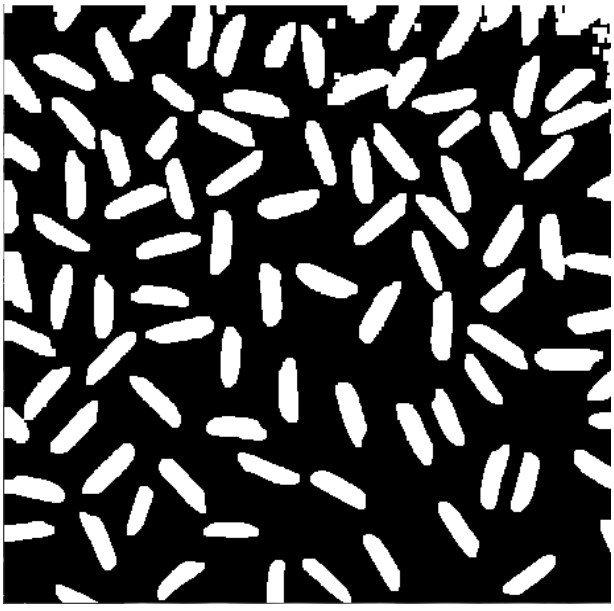
- Ảnh sau khi lọc nhiễu trên miền tần số (bên trái) và sau khi chuyển lại về miền không gian (bên phải):

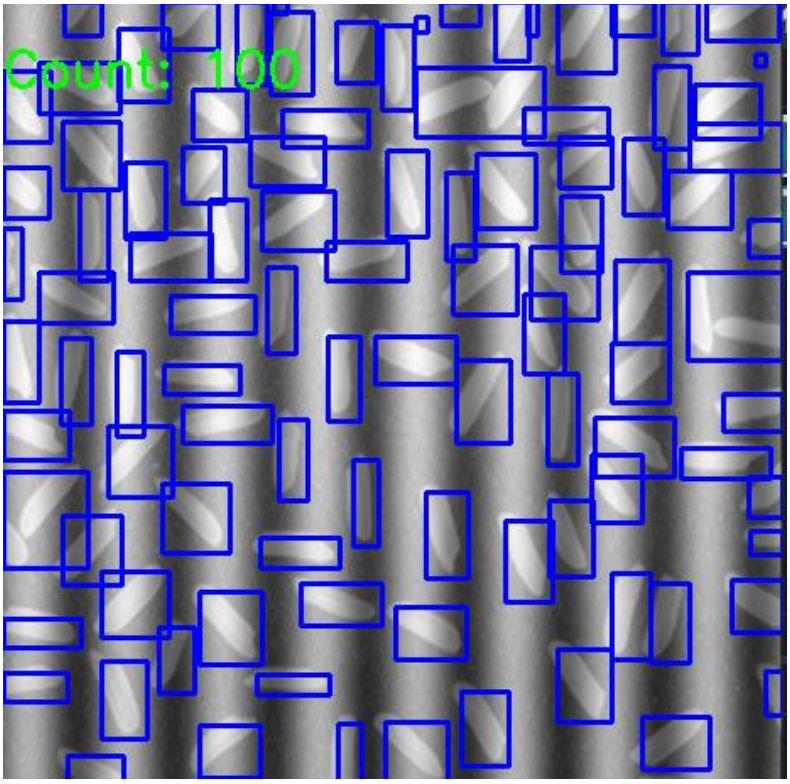
- Ảnh sau khi được chuyển về dạng unit8 (bên trái) và sau khi được lấy nghịch đảo pixel (bên phải)

- Ảnh sau dung adaptive threshold (bên trái) và sau khi sử dụng opening (bên phải):

- Kết quả cuối cùng:



Đối với nhóm ảnh hạt gạo này, có thể thấy trong kết quả có một số các object bị "dính" vào với nhau. Có thể tránh điều này bằng cách sử dụng phép biến đổi erosion (**cv.erode()**) và dilation (**cv.dilate()**) sau khi opening. Tuy nhiên, việc này sẽ làm cho kích thước và vị trí của các object sẽ bị thay đổi đi tương đối lớn.

2.4 Ảnh 5

Đối với ảnh này không cần sử dụng sử dụng median filter để các object có màu gần trùng với màu nền không bị. Ngoài ra cần sử dụng closing thay cho việc sử dụng opening do không có nhiễu để lọc cũng như cần lấp một số các lỗ trên object sau khi sử dụng adaptive threshold.

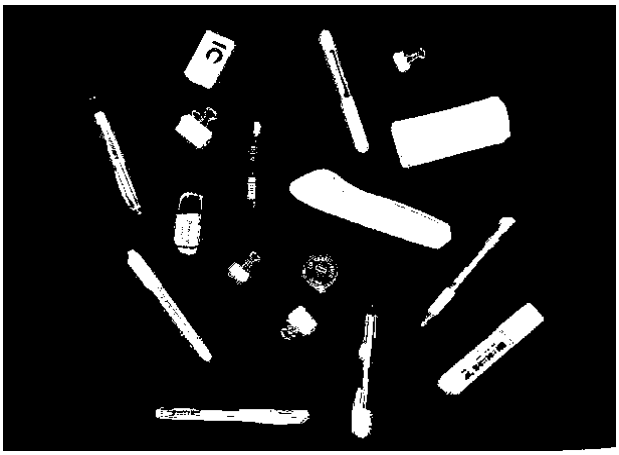
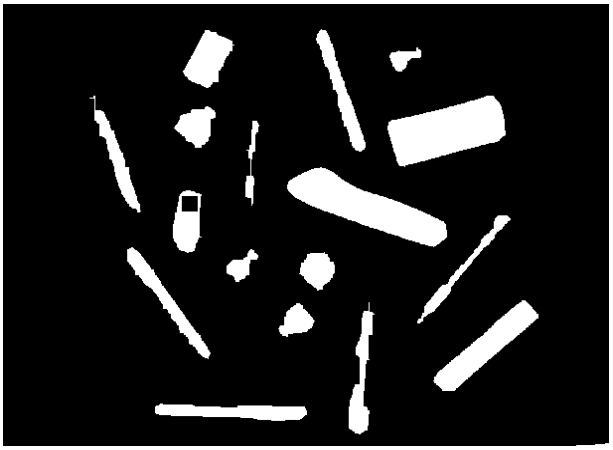
Về chi tiết bộ tham số:

- Adaptive Threshold:
  - Blocksize: 301
  - C: 20
  - Adaptive method: **cv.ADAPTIVE_THRESH_GAUSSIAN_C**
- Closing: closing kernel size 16 x 16

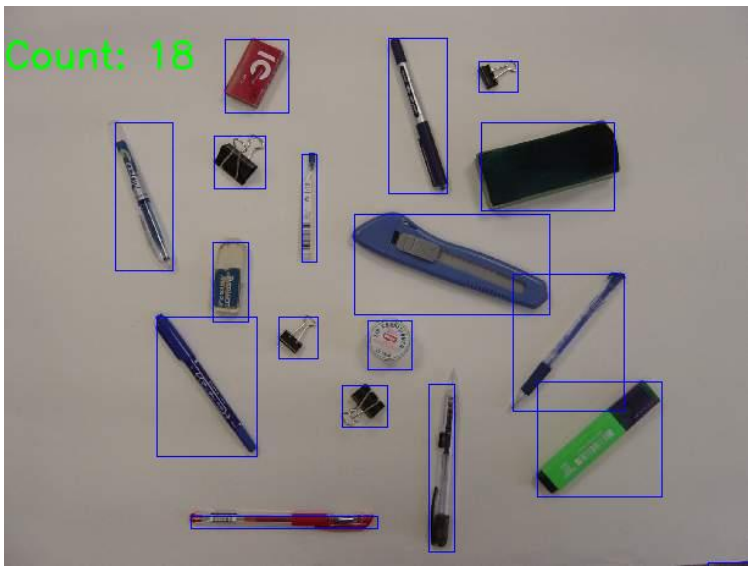
Sau đây là các kết quả xử lý qua từng bước của ảnh 5

- Ảnh gốc (bên trái) và ảnh sau khi chuyển thành ảnh xám (bên phải):


- Sau khi sử dụng adaptive threshold (bên trái) và sau khi sử dụng closing (bên phải):

- Kết quả cuối cùng:



Có thể thấy kết quả cuối cùng đang chưa chính xác tuyệt đối do phát hiện nhầm object ở góc dưới bên phải. Điều này có thể tránh được nếu đưa thêm điều kiện về diện tích tối thiểu của mỗi object. Việc tính toán diện tích của một vùng đã được OpenCV hỗ trợ bằng hàm **cv.contourArea()**.

2.5 Ảnh 6,7

Hai ảnh này sử dụng chung bộ tham số như sau:

- Median filter: median filter kernel: 3x3
- Adaptive Threshold:
    - Blocksize: 401
    - C: 20
    - Adaptive method: **cv.ADAPTIVE_THRESH_GAUSSIAN_C**
- Closing: closing kernel size 16 x 16

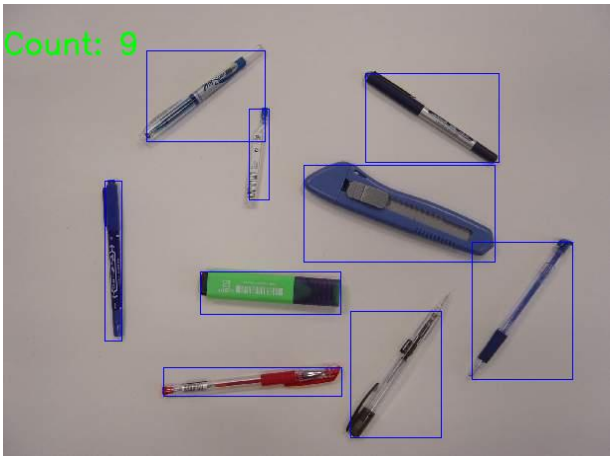
2.5.1 Ảnh 6

Sau đây là các kết quả xử lý qua từng bước của ảnh 6

- Ảnh gốc (bên trái) và ảnh sau khi chuyển thành ảnh xám (bên phải):


- Sau khi sử dụng median filter (bên trái) và sau khi sử dụng adaptive threshold (bên phải):


- Sau khi sử dụng clossing (bên trái) và kết quả cuối cùng (bên phải):

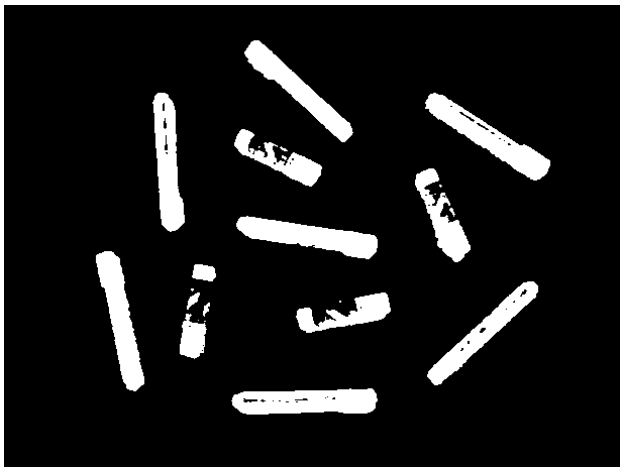



2.5.2 Ảnh 7

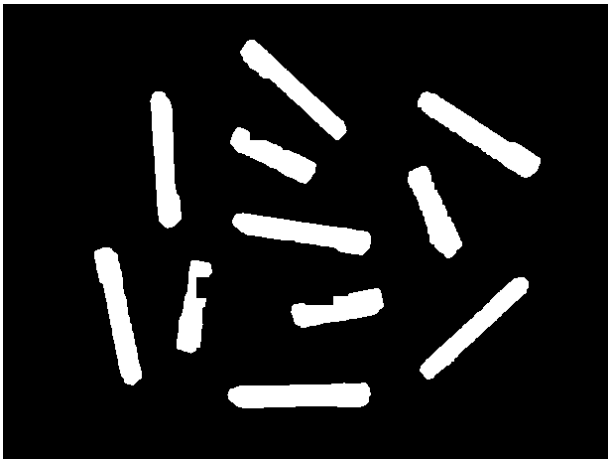
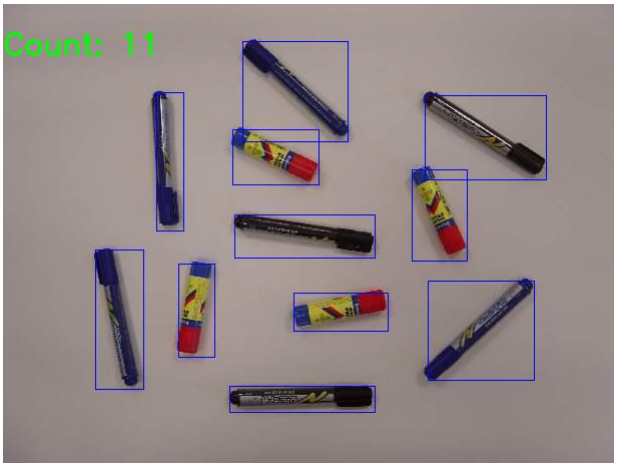
Sau đây là các kết quả xử lý qua từng bước của ảnh 7

- Ảnh gốc (bên trái) và ảnh sau khi chuyển thành ảnh xám (bên phải):


- Sau khi sử dụng median filter (bên trái) và sau khi sử dụng adaptive threshold (bên phải):

- Sau khi sử dụng clossing (bên trái) và kết quả cuối cùng (bên phải):

2.6 Ảnh 8

Với ảnh này không cần sử dụng median filter, do nếu sử dụng sẽ mất bớt một số chi tiết nhỏ, gây tách các object có màu gần tương tự với màu nền.

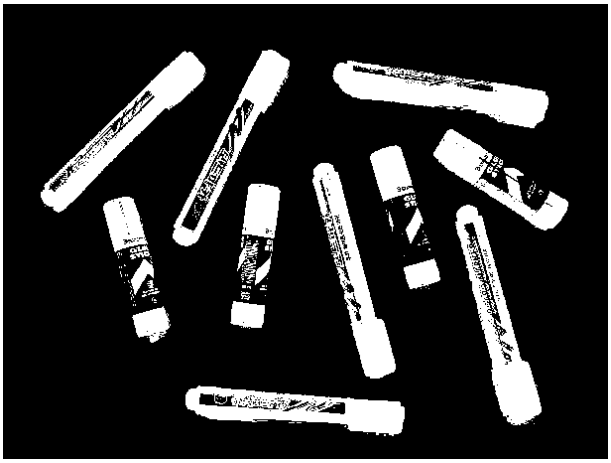
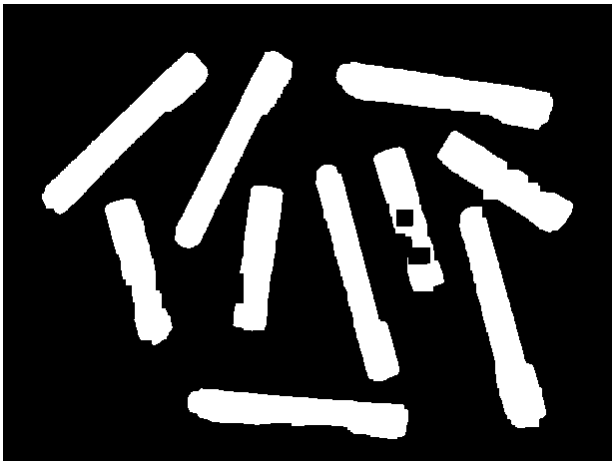
Chi tiết bộ tham số được sử dụng:

- Adaptive Threshold:
    - Blocksize: 401
    - C: 21
    - Adaptive method: **cv.ADAPTIVE_THRESH_GAUSSIAN_C**
- Closing: closing kernel size: 16 x 16

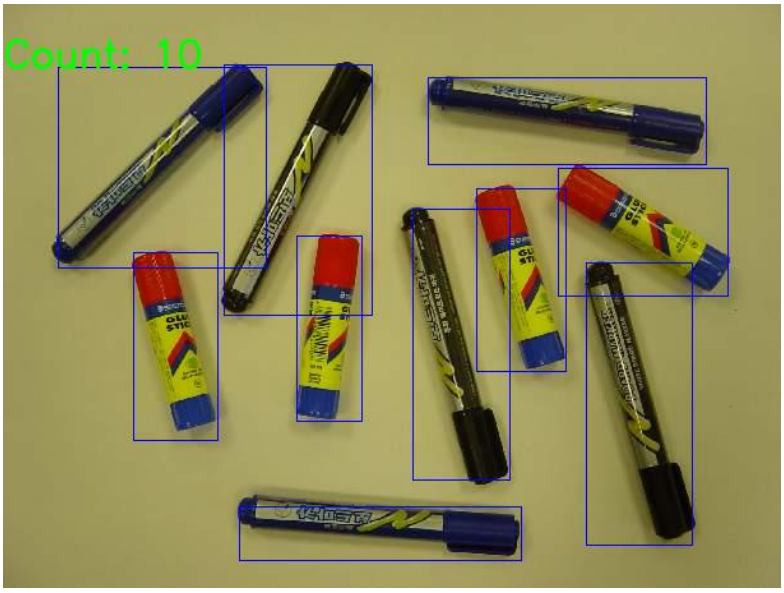
Sau đây là các kết quả xử lý qua từng bước của ảnh 8:

- Ảnh gốc (bên trái) và ảnh sau khi chuyển thành ảnh xám (bên phải):


- Sau khi sử dụng adaptive threshold (bên trái) và sau khi sử dụng closing (bên phải):

- Kết quả cuối cùng:



II. Project 2: nhận dạng đối tượng sử dụng mô hình BoW

1. Giới thiệu về bộ dữ liệu được sử dụng

Bộ dữ liệu được sử dụng ở đây có tên là Nature Image trên Kaggle. Dữ liệu bao gồm 6899 ảnh thuộc 8 lớp khác nhau bao gồm:

- airplane: ảnh máy bay, gồm 727 ảnh.
- car: ảnh các loại xe, gồm 968 ảnh.
- cat: ảnh mèo, gồm 885 ảnh.
- dog: ảnh chó, gồm 702 ảnh.
- flower: ảnh hoa, gồm 843 ảnh.
- fruit: ảnh hoa quả, gồm 1000 ảnh.
- motorbike: ảnh xe máy, gồm 788 ảnh.
- person: ảnh người, gồm 986 ảnh

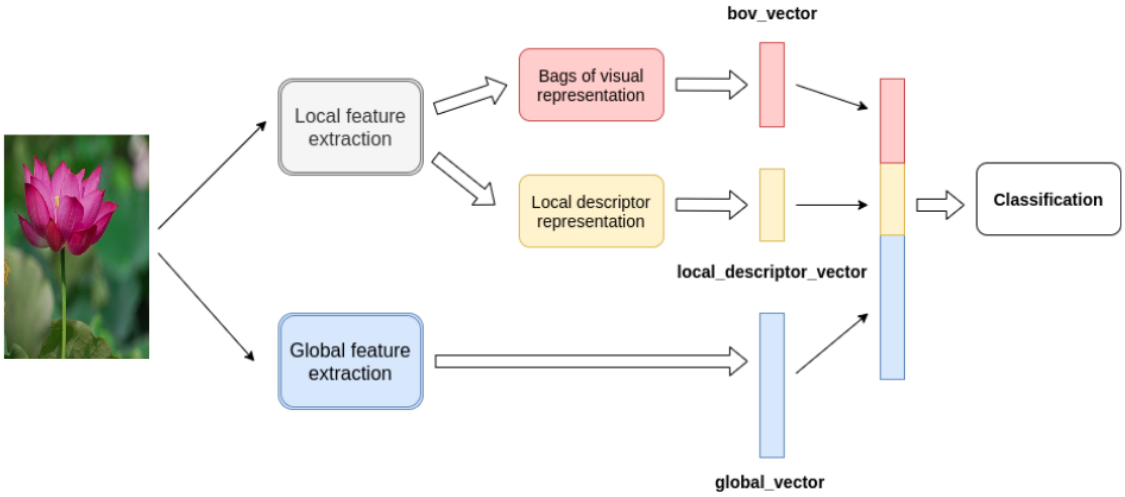
Sau đây là một số ảnh trong tập dữ liệu:

Hình 2: Một số mẫu dữ liệu.

Để thuận thiện cho quá trình huấn luyện cũng như đánh giá mô hình, tiến hành chia tập dữ liệu ban đầu ra làm hai tập train test với tỷ lệ là 8:2. Tỷ lệ này được đảm bảo trên từng nhãn của ảnh.

2. Mô hình đề xuất

2.1 Mô hình tông quan



Hình 3: Kiến trúc tổng quan của mô hình

Mô hình phân loại sẽ sử dụng mô hình bag of visual word để biểu diễn hình ảnh và kết hợp thêm các đặc trưng cục bộ của ảnh và đặc trưng global của ảnh để biểu diễn thông tin của ảnh được tốt hơn. Với đặc trưng cục bộ, mỗi ảnh sẽ được đặt trên một lưới 5x5 và các đặc trưng sẽ được trích xuất trên từng ô rồi kết hợp các đặc ô trên trên các lưới lại với nhau, như vậy mỗi ảnh sẽ mang được đặc trưng cục bộ của từng ô và thông tin vị trí của đặc trưng đó. Với đặc trưng toàn cục sẽ sử dụng các thông tin về histogram và hog để làm giàu thêm thông tin cho biểu diễn. Và cuối cùng các đặc trưng từ mô hình bag of visual word, local feature, global feature sẽ được concat lại với nhau và đưa qua một mô hình phân loại để phân loại ảnh đó là object gì.

2.2 Chi tiết mô hình

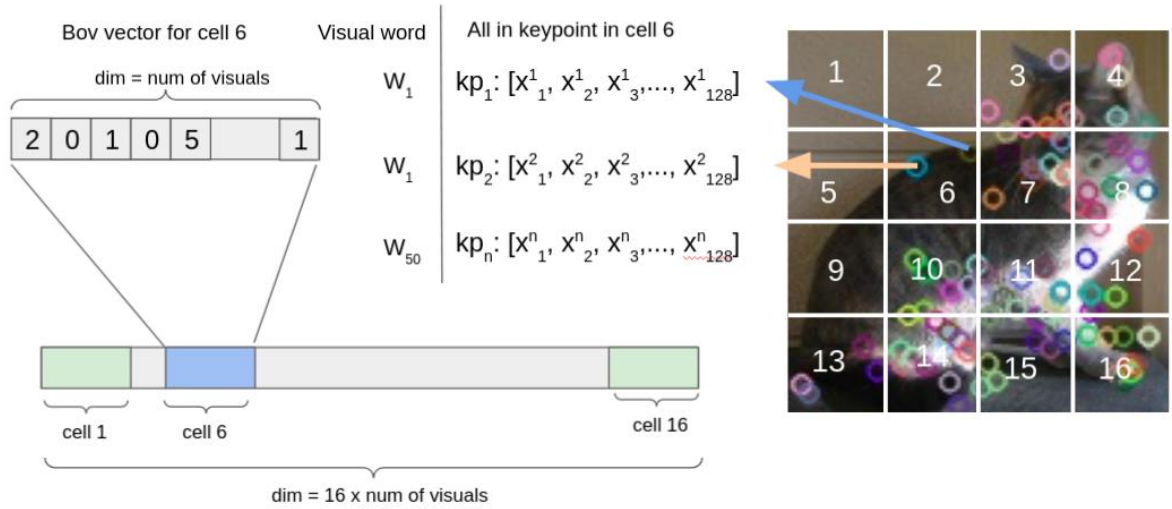
2.2.1 Mô hình bag of visual word

Mô hình phân loại dựa trên mô hình bag of visual gồm 3 bước sau:

- Feature Extraction: trích xuất các descriptors là các vector đặc trưng của từng ảnh đối với tất cả các ảnh.
- Visual Word Vocabulary Construction: xây dựng bộ từ điển visual word bằng cách phân cụm các vector descriptors sử dụng thuật toán kMean.
- Image represent by bag of visual word: biểu diễn hình ảnh bằng bag of visual words dưới dạng tần suất xuất hiện của các từ.

2.2.2 Local feature

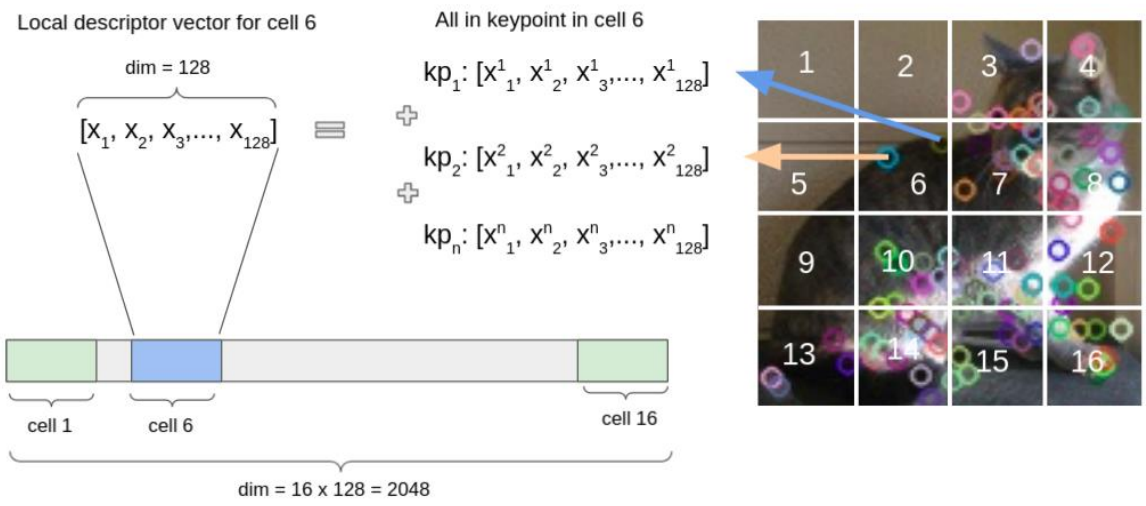
Với phương pháp bag of visual thông thường, thông tin kết cấu lại khó biểu diễn tổng thể bởi nó không quan tâm đến vị trí của các descriptor. Để giải quyết vấn đề đó nhóm đã đưa ý tưởng biểu diễn ảnh trên từng vùng để có thể thêm thông tin về vị trí của các descriptor.



Hình 4: Sử dụng local bag of visual words để biểu diễn thông tin vị trí của descriptor.

Với mỗi ảnh đầu vào, sẽ sử dụng một lưới nxn đặt lên ảnh, như ví dụ minh họa trên sử dụng lưới 4x4 và như vậy ảnh sẽ được chia thành 16 vùng cục bộ tương ứng với 16 cell trên lưới.. Khi thực hiện bước xây dựng vector biểu diễn bag of visual word cho ảnh, ngoài việc xây dựng trên toàn bộ ảnh, thì sẽ xây dựng thêm trên từng ô (cell) của lưới. Như hình trên có 16 cells thì sẽ có 16 vector bag of visual words tương ứng với từng ô trên lưới và cuối cùng concat chúng lại thu được vector biểu diễn local bag of visual words cho ảnh. Như vậy với các biểu diễn như này ta sẽ biết được có những visual word nào trong ảnh xuất hiện và xuất hiện ở vị trí nào của ảnh, những visual word nào thường xuất hiện gần nhau khi chúng cùng xuất hiện trong một vùng và xuất hiện tại vùng lân cận.

Một vấn đề khi việc sử dụng bag of word ta chỉ biết được các visual word xuất hiện trong ảnh hay không và xuất hiện bao nhiêu lần, chứ không biết được cụ thể chi tiết đặc trưng đó là gì. Ví dụ giả sử 2 ảnh chó và mèo đều xuất hiện visual word biểu diễn cho "**tai**", thu nhiên thực tế một số loại chó có **tai cụp** khác với tai mèo, nếu sử dụng bag of word thì đặc trưng đó bị coi giống nhau. Hơn thế nữa, việc xây dựng bộ từ điển visual word cũng không đảm bảo rằng 2 vector descriptor cũng biểu diễn cho một đặc trưng và nó phụ thuộc vào số từ mà chúng ta mong muốn. Nếu sử dụng số lượng visual word ít thì khả năng biểu diễn kém hơn. Để giải quyết này và tận dụng ý tưởng của local bag of visual words nhóm đã đưa ra ý tưởng thay vì sử dụng bag of words để biểu diễn đặc trưng local của từng cell, ta sẽ sử dụng luôn các vector đặc trưng của các vùng (mỗi vùng sẽ là một ô trên lưới). Vector biểu diễn đặc trưng cho một vùng sẻ bằng tổng các vector descriptor xuất hiện trong vùng. Minh họa cách biểu diễn như hình sau:



Hình 5: Sử dụng local feature để biểu diễn thông tin vị trí của descriptor.

Như hình trên ta thấy ảnh con mèo sẽ được chia thành 16 vùng khác nhau bằng cách sử dụng một lưới 4x4. Sau khi đưa qua bộ trích xuất đặc trưng ta sẽ có các keypoint và các descriptor tương ứng của các keypoint. Dựa vào keypoint ta sẽ biết được vị trí của descriptor và xác định nó thuộc vùng nào trong ảnh. Với mỗi vùng trong ảnh sẽ tính tổng các descriptor lại và cuối cùng concat tất cả các vector đặc trưng của tất cả các vùng thu được vector local feature descriptor (hay local descriptor).

2.2.3 Global feature

Nếu chỉ sử dụng các đặc trưng cục bộ - local feature thì sẽ không biểu diễn được một số thông tin về màu sắc cũng như kết cấu của ảnh. Các thông tin này tương đối quan trọng, có thể giúp tăng kết quả phân loại. Một vài ví dụ điển hình để thấy rằng các thông tin này quan trọng đó là:

- Có rất ít quả có màu xanh dương
- Phân biệt chó và mèo cần dựa trên các thông tin về khuôn mặt của chúng
- …

Chính vì thể, cần sử dụng thêm các đặc trưng toàn cục – global feature. Các đặc trưng toàn cục được sử dụng để thử nghiệm trong mô hình này đó là:

- Histogram of oriented gradients – HoG.
- Histogram của ảnh cho 3 channel RGB.

3. Kết quả thực nghiệm

- Sau đây là kết quả thực nghiệm với mô hình bag of visual word cơ bản.

Num of visuals	Accuracy	Precision	Recall	F1-Score
50	0.6377	0.6258	0.6247	0.6245
100	0.6876	0.6773	0.6756	0.6762
200	0.7187	0.7069	0.7064	0.7062
300	0.7237	0.7130	0.7125	0.7125
400	0.7483	0.7391	0.7402	0.7393
500	0.7324	0.7232	0.7226	0.7226

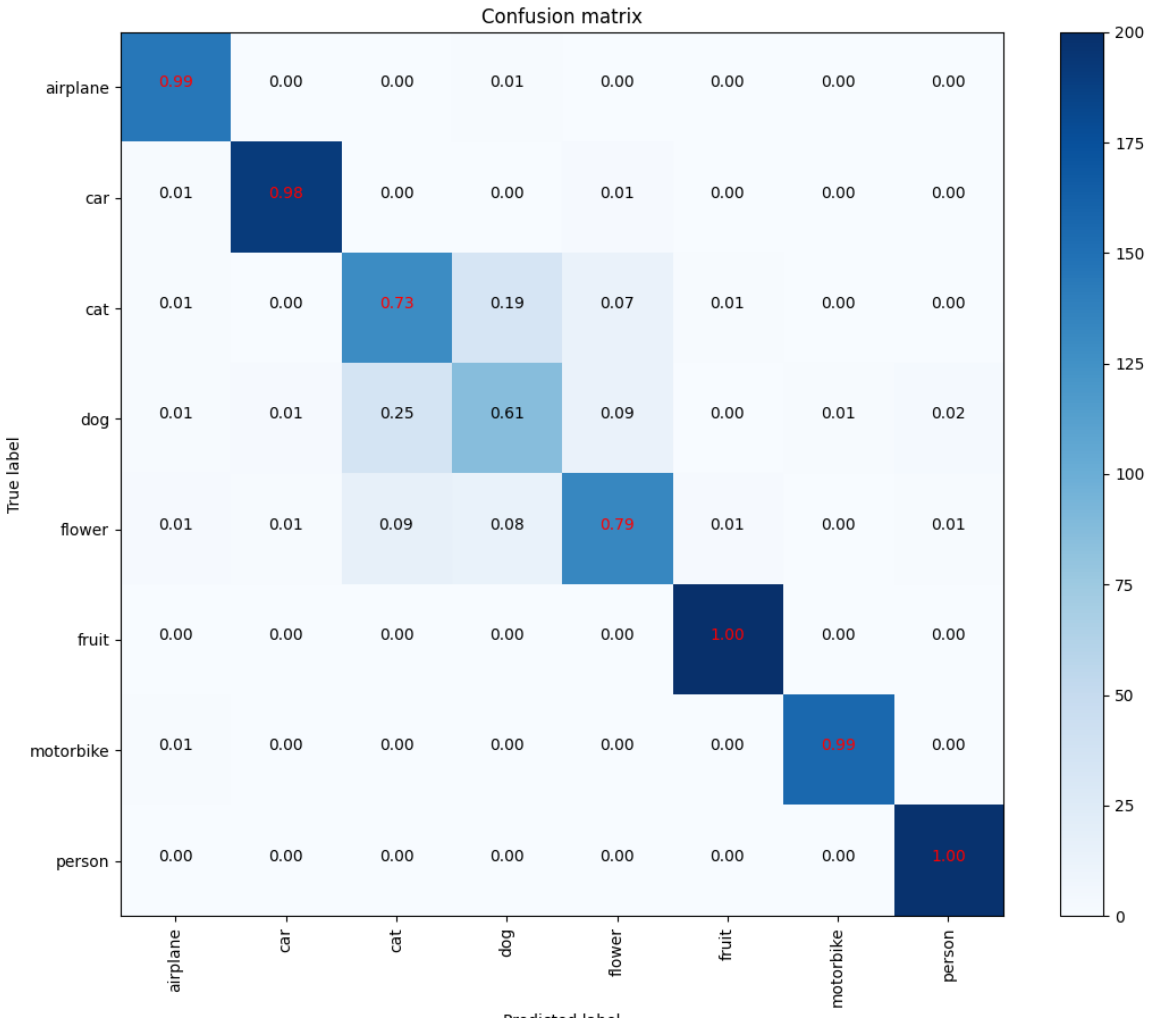
- Kết quả thực nghiệm sử dụng mô hình local bags of word và local feature descriptor (Num of visual=400: số lượng từ trong từ điển là 400)

Model	Accuracy	Precision	Recall	F1-Score
BoW	0.7483	0.7391	0.7402	0.7393
BoW + local bag of word	0.8055	0.8055	0.8027	0.8035
BoW + local feature descriptor	0.8040	0.8040	0.7924	0.7924
BoW + local bag of word + local feature descriptor	0.8011	0.7897	0.7901	0.7897

- Kết quả thực nghiệm sử dụng mô hình BoW kết hợp với global feature và local feature:

Model	Accuracy	Precision	Recall	F1-Score
BoW	0.7483	0.7391	0.7402	0.7393
BoW + local feature descriptor + histograms	0.8365	0.8292	0.8292	0.8277
BoW + local feature descriptor + HoG	0.8886	0.8803	0.8798	0.8791
BoW + local feature descriptor + histograms + HoG	0.8958	0.8860	0.8871	0.8864
BoW + local BoW + histograms	0.8459	0.8408	0.8369	0.8385
BoW + local BoW + HoG	0.9009	0.8928	0.8935	0.8931
BoW + local BoW + histograms + HoG	0.9052	0.8984	0.8997	0.8988
Basic CNN	0.9178	0.9101	0.9106	0.9102

- Kết quả chi tiết cho mô hình BoW + local BoW + histograms + HoG:



Hình 6: Kết quả chi tiết cho mô hình BoW + local BoW + histograms + HoG.

4. Kết luận

Có thể thấy các mô hình sử dụng cả global feature và local feature cho kết quả tốt nhất trong số các mô hình bag of visual word, tuy nhiên vẫn còn kém so với việc sử dụng mô hình mạng tích chập cơ bản để phân loại. Kết quả khi sử dụng local bag of word có cao hơn so với việc sử dụng local feature descriptor. Điều này là do trong local feature descriptor tiến hành cộng các descriptor trong cùng một sô trong lưới lại dẫn đến việc mất mát nhiều thông tin hơn.

Trong tương lai, chúng em sẽ tiếp tục thử nghiệm mô hình dùng cả local BoW, local feature descriptor, histogram và HoG. Cùng với đó là sử dụng các local feature extraction khác như SURF, ORB, ... cũng như các global feature khác như invariant moments, PHOG, Co-HOG, ... Ngoài ra, để cải thiện kết quả, chúng em sẽ sử dụng thêm dữ liệu của các tập dog, cat và flower để huấn luyện thêm do kết quả test ở cả 3 tập này đều thấp. Cùng với đó là tiến hành huấn luyện lại mô hình với các tập dữ liệu lớn hơn như CIFAR10, COIL100,…

III. Tài liệu tham khảo

Slide môn học Thị giác máy tính – IT5409

LEARN OPENCV in 3 HOURS with Python | Including 3xProjects | Computer Vision: https://www.youtube.com/watch?v=WQeoO7MI0Bs&t=5849s

Nature Image Dataset: https://www.kaggle.com/prasunroy/natural-images

Bag of Visual Words: https://github.com/gurkandemir/Bag-of-Visual-Words