TRƯỜNG ĐẠI HỌC THỦY LỢI

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN**

HỌC PHẦN:TÍNH TOÁN MỀM

**ĐỀ TÀI:**

**Phát hiện tàu biển từ ảnh vệ tinh dựa trên phương pháp bán giám sát mờ an toàn**

*Giảng viên hướng dẫn: Trần Mạnh Tuấn*

Nhóm sinh viên thực hiện:

1. Nguyễn Ngọc Bách - 2151163668

2. Trương Quốc Bảo - 2151163669

3. Lê Mạnh Duy - 2151163677

4. Nguyễn Văn Nam - 2151160525

**Hà Nội, năm 2024**

**MỤC LỤC**

[**I. MÔ TẢ BÀI TOÁN 5**](#_ddc3gh3sapj5)

[A. Lý do chọn đề tài 5](#_p2nk5wq4m2xx)

[B. Mục tiêu nghiên cứu 5](#_bh3hcwyrdd81)

[1.1 Mục tiêu tổng quát 5](#_gt2u3yi6f1x5)

[1.2 Mục tiêu cụ thể 5](#_afx76wws0i9c)

[C. Phương pháp, dữ liệu và nội dung nghiên cứu 5](#_auwiqekkaz53)

[2.1 Phương pháp nghiên cứu 5](#_4xkj31hhosjg)

[2.2 Nội dung nghiên cứu 6](#_gpc470eygexm)

[D. Phân tích dữ liệu 6](#_uq8aqvijp14j)

[3.1 Dữ liệu ảnh 6](#_sebxz9o9s233)

[3.2 Thông tin về dữ liệu: 6](#_v4jqosk8jdt2)

[3.3 Cấu trúc của bộ dữ liệu: 6](#_v90xy34vaz8k)

[3.4 Đặc điểm của dữ liệu: 7](#_criaaam566il)

[3.5 Chuẩn bị dữ liệu: 7](#_8vhm8oa9f4o)

[**II. Phương pháp và các thuật toán bán giám sát mờ 12**](#_i3w8eju3u7jc)

[1.1. Vấn đề phân cụm dữ liệu với độ tin cậy 12](#_r8b4jy4cj3w7)

[1.2. Thuật toán phân cụm mờ tiêu chuẩn FCM (Fuzzy C-Mean) 13](#_gtbdtnze9fj5)

[1.3. Phân cụm mờ bán giám sát tiêu chuẩn SSFCM (Semi-Supervised Fuzzy C-Means) 14](#_2x722y58vtea)

[1.4. Phương pháp TS3FCM 15](#_jmgskkea19p1)

[1.5. Thuật toán phân cụm bán giám sát mờ an toàn có trọng số tin cậy (CS3FCM) 18](#_kc6pyt2pynhe)

[**III. CODE GIẢI THUẬT BÁN GIÁM SÁT MỜ PYTHON 21**](#_8zbn537bvbik)

[A. Thuật toán FCM 21](#_ujtjwcjmb5v9)

[1.1. Các bước giải thuật toán FCM 21](#_nls62lm0eu2o)

[1.2. Code thuật toán FCM 22](#_ppf5kmjmpbho)

[1.3. Code Thuật toán FCM cải tiến 24](#_ij9mt3hhkmt5)

[B. Thuật toán SSFCM 27](#_wsn1ok2t5rbz)

[2.1. Các bước giải thuật toán SSFCM 27](#_2diod51e57g3)

[2.2. Code thuật toán SSFCM 27](#_1necewn52vep)

[C. Thuật toán TS3FCM 29](#_jn4kf6pnb3a7)

[3.1. Các bước giải thuật toán TS3FCM 29](#_cag8ie66r1wp)

[3.2. Code thuật toán TS3FCM 32](#_u0gzn0hvmfsc)

[D. Thuật toán CS3FCM 34](#_hrtwl4al53lr)

[4.1. Các bước giải thuật toán CS3FCM 34](#_o4ddbtkrj21r)

[4.2. Code thuật toán CS3FCM 35](#_axwi5y9zgk0s)

[E. Code Tổng hợp 36](#_m5nhwsm1l8do)

[5.1. Thư viện cần dùng 36](#_1kl55hm4z9g)

[5.2 Main 37](#_yivjs21dpd5b)

[5.3 Gui ( Giao diện) 41](#_8p19gs6nij7)

[**IV. KẾT QUẢ 47**](#_aaqu1pwtxy36)

[4.1 Main 47](#_rx2ketv3g10w)

[4.2 Gui ( giao diện ) 48](#_ni92dzc5tqrt)

[**V. KẾT LUẬN 49**](#_lhezrl1r0fth)

[**TÀI LIỆU THAM KHẢO 50**](#_yc6d9w9sign)

**BẢNG PHÂN CHIA CÔNG VIỆC**

| **Người phụ trách** | **Công việc** |
| --- | --- |
| Nguyễn Ngọc Bách | * Nhóm trưởng * FCM * main.py * CS3FCM |
| Nguyễn Văn Nam | * Báo cáo * FCM cải tiến * TS3FCM |
| Trương Quốc Bảo | * Phương pháp và công cụ xử lý ảnh * SSFCM |
| Lê Mạnh Duy | * Xử lý ảnh * Giao diện |

# I. MÔ TẢ BÀI TOÁN

## **A. Lý do chọn đề tài**

Trong bối cảnh thế giới ngày càng phụ thuộc vào công nghệ và dữ liệu lớn, việc theo dõi và quản lý các hoạt động hàng hải trở nên quan trọng hơn bao giờ hết. An ninh biển, quản lý giao thông hàng hải, và bảo vệ môi trường biển là những lĩnh vực then chốt cần sự hỗ trợ của các công nghệ tiên tiến. Trong môi trường đầy thử thách này, việc phát hiện và giám sát tàu biển từ ảnh vệ tinh đã trở thành một nhu cầu cấp thiết. Đề tài "**Phát hiện tàu biển từ ảnh vệ tinh dựa trên phương pháp bán giám sát mờ an toàn**" không chỉ cung cấp một phương pháp tiếp cận mới mẻ mà còn mang lại những giải pháp thực tiễn, góp phần quan trọng vào việc nâng cao hiệu quả và an toàn cho các hoạt động trên biển. Việc nghiên cứu và triển khai đề tài này sẽ giúp chúng em phát triển kỹ năng giải quyết các vấn đề phức tạp, đồng thời đóng góp vào sự phát triển bền vững của ngành hàng hải trong kỷ nguyên số.

## **B. Mục tiêu nghiên cứu**

### 1.1 Mục tiêu tổng quát

- Nghiên cứu một số phương pháp phân cụm bán giám sát mờ an toàn ứng dụng phát hiện tàu biển từ ảnh vệ tinh trong bài toán quản lý cảng biển.

### 1.2 Mục tiêu cụ thể

• Mục tiêu 1 : Nghiên cứu phương pháp phân cụm bán giám sát mờ mới cho phân vùng dữ liệu với độ tin cậy.

• Mục tiêu 2 : Nghiên cứu phương pháp phân cụm bán giám sát mờ mới trên tập mờ viễn cảnh cho phân vùng dữ liệu với độ tin cậy và dữ liệu nhiễu.

• Mục tiêu 3 : Nghiên cứu phương pháp phân cụm bán giám sát mờ an toàn mới với nhiều tham số mờ cho phân vùng dữ liệu với độ tin cậy và dữ liệu nhiễu.

• Mục tiêu 4 : Ứng dụng các phương pháp để phát hiện tàu biển từ ảnh vệ tinh trong bài toán cảng biển.

## **C. Phương pháp, dữ liệu và nội dung nghiên cứu**

### 2.1 Phương pháp nghiên cứu

- Phương pháp nghiên cứu lý thuyết: Tổng hợp và nghiên cứu các tài liệu liên quan đến phân cụm mờ, phân cụm bán giám sát mờ. Tìm hiểu các hướng nghiên cứu mới liên quan đến phân cụm bán giám sát mờ và đề xuất phương pháp cải tiến.

- Phương pháp nghiên cứu thực tiễn: Tìm hiểu bài toán quản lý cảng biển để thấy rằng phát hiện tàu trên biển đóng một vai trò quan trọng trong quản lý cảng, bao gồm giao thông tàu, cứu hộ hàng hải, vận chuyển hàng hóa và xây dựng hệ thống an ninh quốc phòng.

### 2.2 Nội dung nghiên cứu

• Nghiên cứu,phương pháp phân cụm bán giám sát mờ an toàn trong phân vùng dữ liệu có độ tin cậy.

• Nghiên cứu, phát triển phương pháp phân cụm bán giám sát mờ mở rộng trên tập mờ viễn cảnh. Kiểm chứng, so sánh hiệu quả so với một số thuật toán liên quan khác.

• Nghiên cứu, phát triển phương pháp phân cụm bán giám sát mờ an toàn với nhiều tham số mờ để xử lý dữ liệu nhiễu.

• Ứng dụng trong bài toán phát hiện tàu biển từ ảnh vệ tinh cảng biển.

## **D. Phân tích dữ liệu**

### 3.1 Dữ liệu ảnh

Dữ liệu ảnh vệ tinh được sử dụng trong dự án "Phát hiện tàu biển từ ảnh vệ tinh dựa trên phương pháp bán giám sát mờ an toàn" được lấy từ cuộc thi "Airbus Ship Detection" trên nền tảng Kaggle. Dữ liệu này gồm các ảnh vệ tinh có độ phân giải cao, trong đó các tàu biển được chú thích (annotate) rõ ràng, giúp cho việc huấn luyện và kiểm thử các mô hình phát hiện tàu biển.

### 3.2 Thông tin về dữ liệu:

- Nguồn dữ liệu:

* Cuộc thi "Airbus Ship Detection" trên Kaggle.
* URL: Airbus Ship Detection

- Định dạng dữ liệu:

* Dữ liệu gồm các ảnh vệ tinh định dạng .jpg hoặc .png.
* Các chú thích (annotations) được cung cấp dưới dạng file CSV, trong đó mỗi dòng tương ứng với một hộp giới hạn (bounding box) xác định vị trí của một tàu biển trong ảnh.

### 3.3 Cấu trúc của bộ dữ liệu:

- Thư mục train: Chứa các ảnh vệ tinh dùng cho việc huấn luyện mô hình.

- Thư mục test: Chứa các ảnh vệ tinh dùng để kiểm thử mô hình.

- File train\_ship\_segmentations\_v2.csv: Chứa các chú thích về vị trí của tàu biển trong các ảnh huấn luyện. Mỗi dòng trong file này bao gồm các thông tin như ID của ảnh và tọa độ của hộp giới hạn bao quanh tàu biển.

- File sample\_submission\_v2.csv: Mẫu file cho kết quả dự đoán, bao gồm ID của ảnh và các tọa độ của hộp giới hạn được dự đoán.

### 3.4 Đặc điểm của dữ liệu:

- Số lượng ảnh:

* Bộ dữ liệu huấn luyện: Hàng chục ngàn ảnh.
* Bộ dữ liệu kiểm thử: Hàng ngàn ảnh.

- Kích thước ảnh:

* Độ phân giải của ảnh có thể khác nhau, nhưng đa phần đều có độ phân giải cao để đảm bảo chi tiết.

- Chú thích:

* Mỗi ảnh có thể chứa từ 0 đến nhiều tàu biển.
* Tọa độ của các hộp giới hạn được cung cấp dưới dạng chuỗi RLE (Run-Length Encoding), giúp tiết kiệm dung lượng lưu trữ và dễ dàng xử lý.

- Phân tích dữ liệu ban đầu:

* Phân phối kích thước tàu: Qua phân tích sơ bộ, kích thước của các tàu biển trong ảnh có thể dao động lớn, từ các tàu nhỏ đến các tàu hàng lớn. Việc này đòi hỏi mô hình phải linh hoạt trong việc phát hiện các đối tượng với kích thước khác nhau.

- Môi trường xung quanh: Các tàu biển xuất hiện trong nhiều bối cảnh khác nhau, từ các vùng biển mở rộng đến các cảng biển đông đúc, ảnh hưởng đến độ khó của bài toán phát hiện.

- Tính không đồng nhất: Các ảnh vệ tinh có thể khác nhau về điều kiện ánh sáng, thời tiết, và góc chụp, tạo ra một thách thức lớn cho việc phát hiện chính xác tàu biển.

### 3.5 Chuẩn bị dữ liệu:

- Chọn lọc 30 ảnh có nhiều tàu và biển: chọn lọc ảnh từ tập test\_v2 của "Airbus Ship Detection" dữ liệu trên kaggle. <https://www.kaggle.com/competitions/airbus-ship-detection/data?select=test_v2>

- Điều chỉnh kích thước ảnh về 201x201: Sử dụng code python để điều chỉnh các ảnh về cùng một dạng kích thước 201 x 201 để dễ dàng xử lý.

* Code python về thay đổi kích thước ảnh:

import os

from PIL import Image

# Đường dẫn tới thư mục chứa ảnh gốc

input\_folder = 'Ship1'

# Đường dẫn tới thư mục để lưu ảnh đã thay đổi kích thước

output\_folder = 'ship'

# Tạo thư mục đầu ra nếu nó chưa tồn tại

if not os.path.exists(output\_folder):

os.makedirs(output\_folder)

# Duyệt qua tất cả các tệp trong thư mục đầu vào

for filename in os.listdir(input\_folder):

if filename.endswith(('.png', '.jpg', '.jpeg', '.bmp', '.gif')):

# Đường dẫn tới tệp gốc

input\_path = os.path.join(input\_folder, filename)

# Đường dẫn tới tệp đầu ra

output\_path = os.path.join(output\_folder, filename)

# Mở ảnh

with Image.open(input\_path) as img:

# Thay đổi kích thước ảnh

img\_resized = img.resize((201, 201))

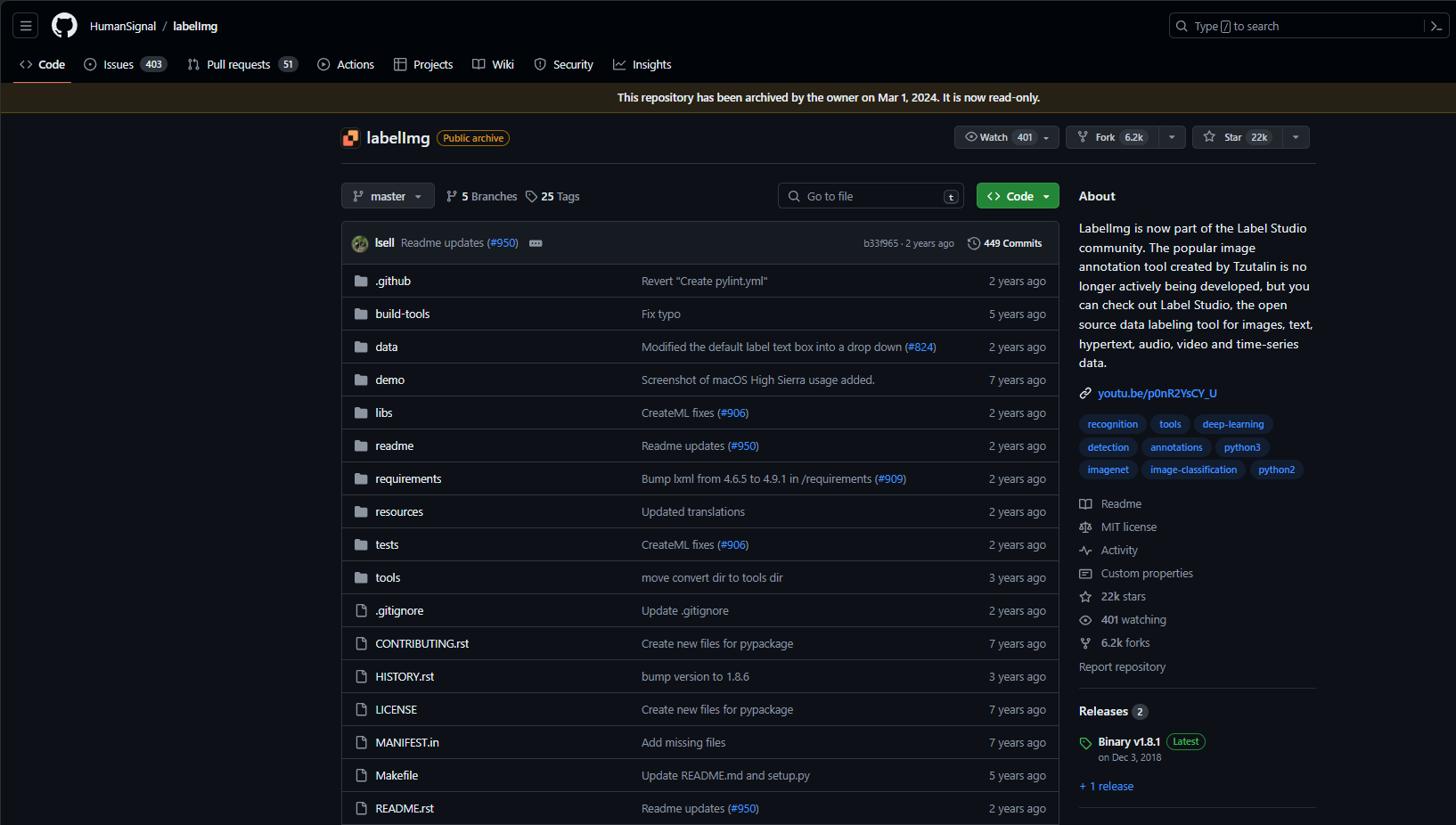
# Lưu ảnh đã thay đổi kích thước vào thư mục đầu ra

img\_resized.save(output\_path)

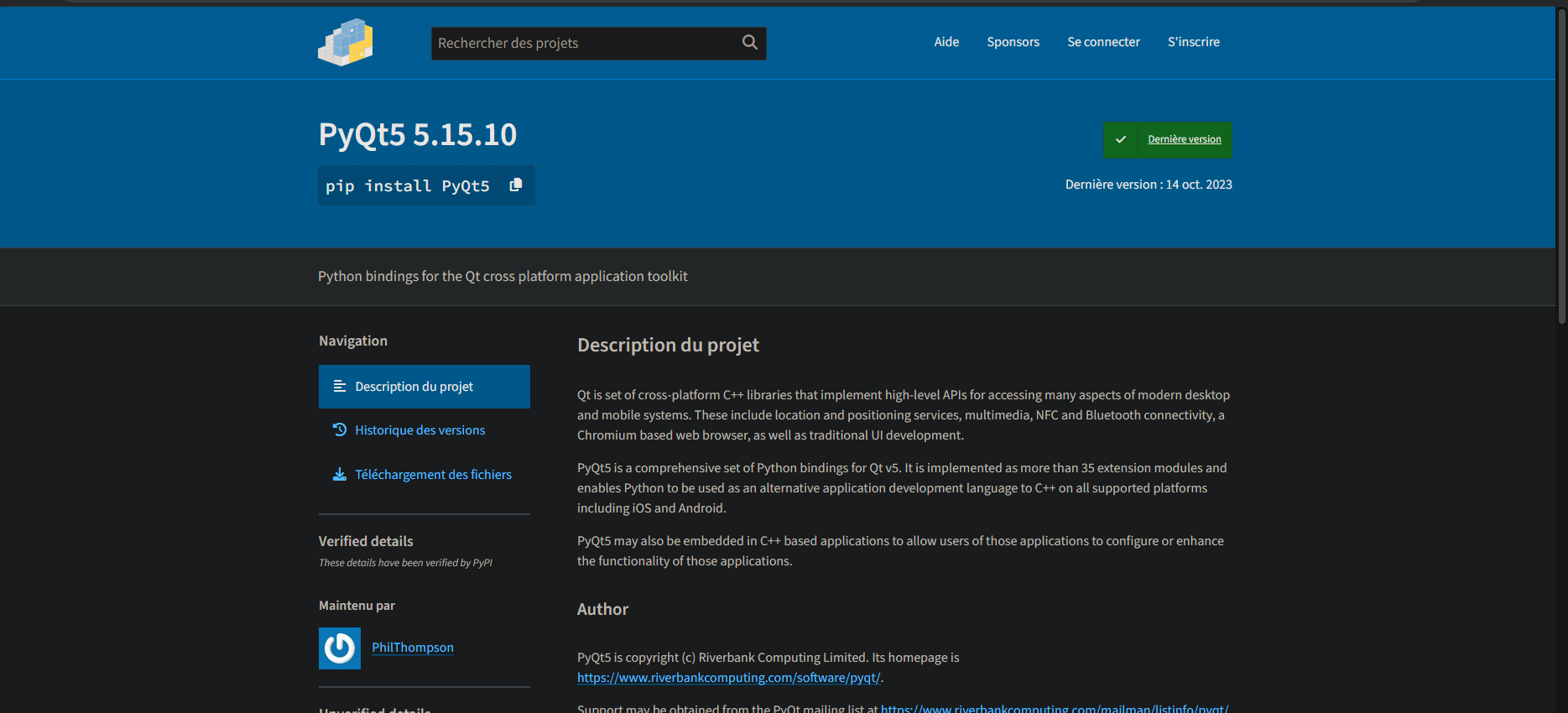
print("Đã hoàn thành việc thay đổi kích thước và lưu ảnh vào thư mục mới.")

- Chọn 15 ảnh để gán nhãn: sử dụng công cụ labelImg được tải từ github, python, PyQt5 và lxm để thực hiện xử lý và gán nhãn cho ảnh.

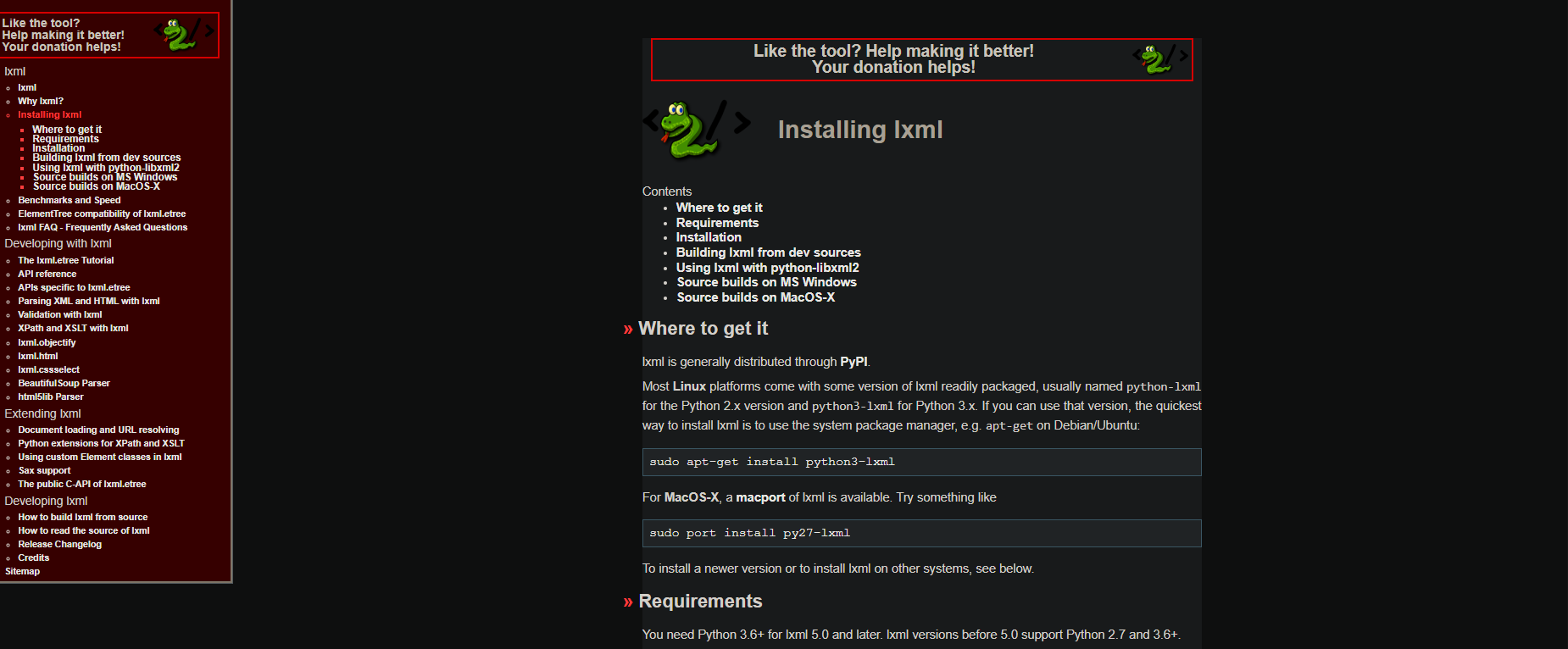
* Tải labelImg: <https://github.com/HumanSignal/labelImg?tab=readme-ov-file>



* Tải PyQt5: <https://pypi.org/project/PyQt5/>



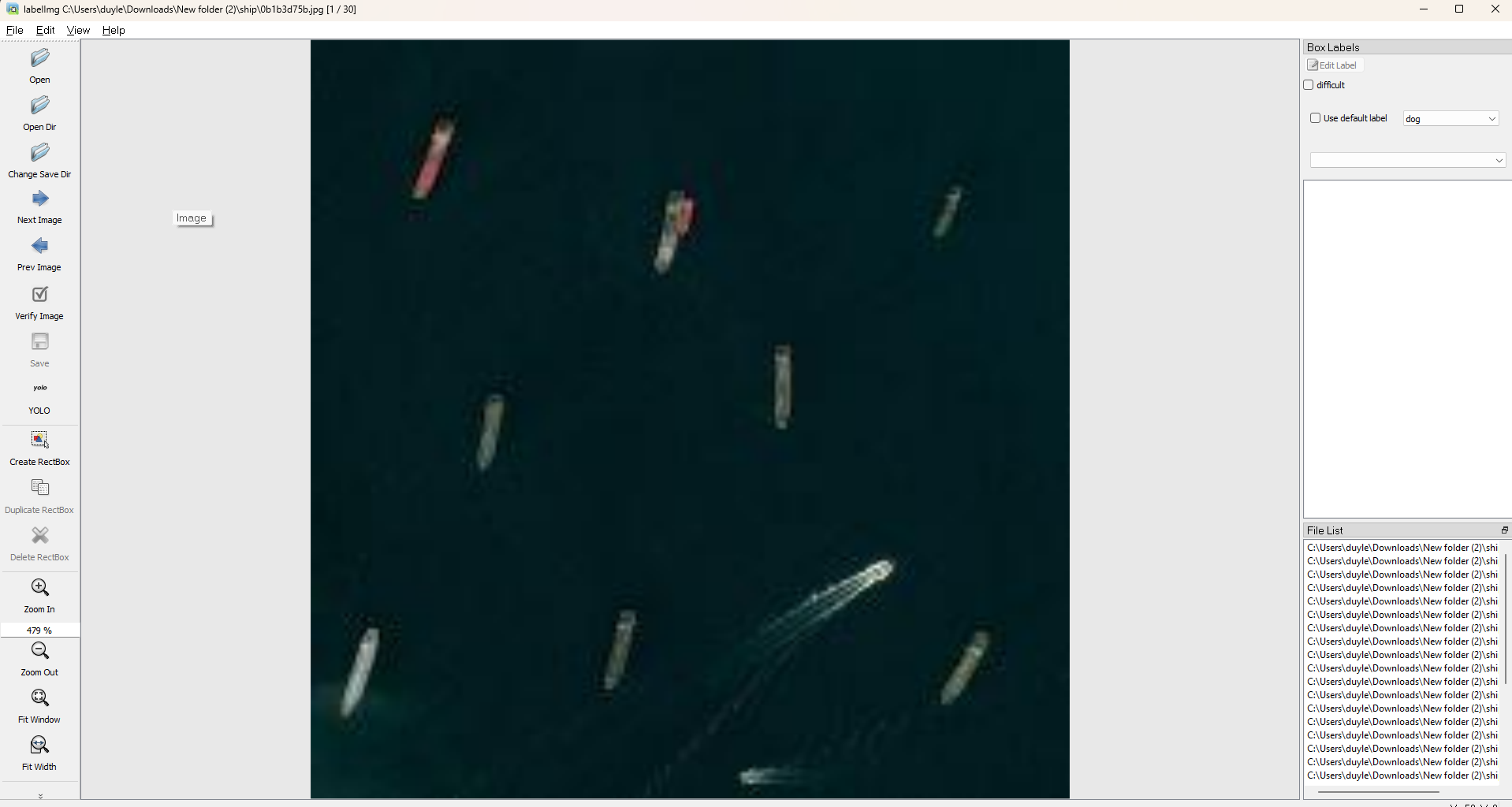
* Tải lxml: <https://lxml.de/installation.html>



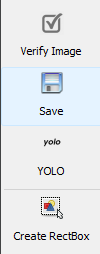
* B1: Vào thư mục chứa labelImg, mở trong terminal và thực hiện chạy lệnh:   
   pyrcc5 -o libs/resources.py resources.qrc

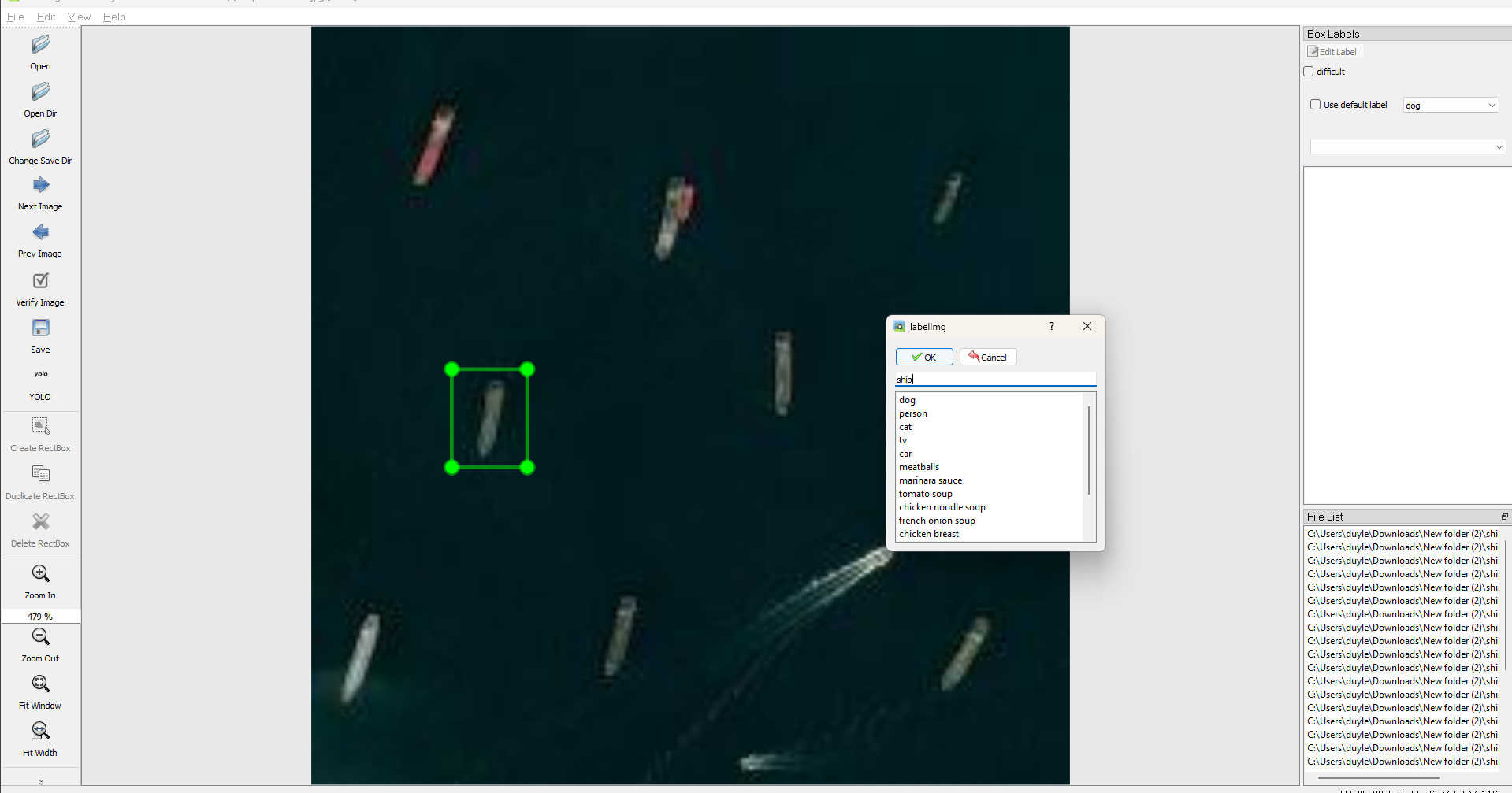
python labelImg.py

* B2:Sau khi vào được phần mềm, mở thư mục chứa ảnh:

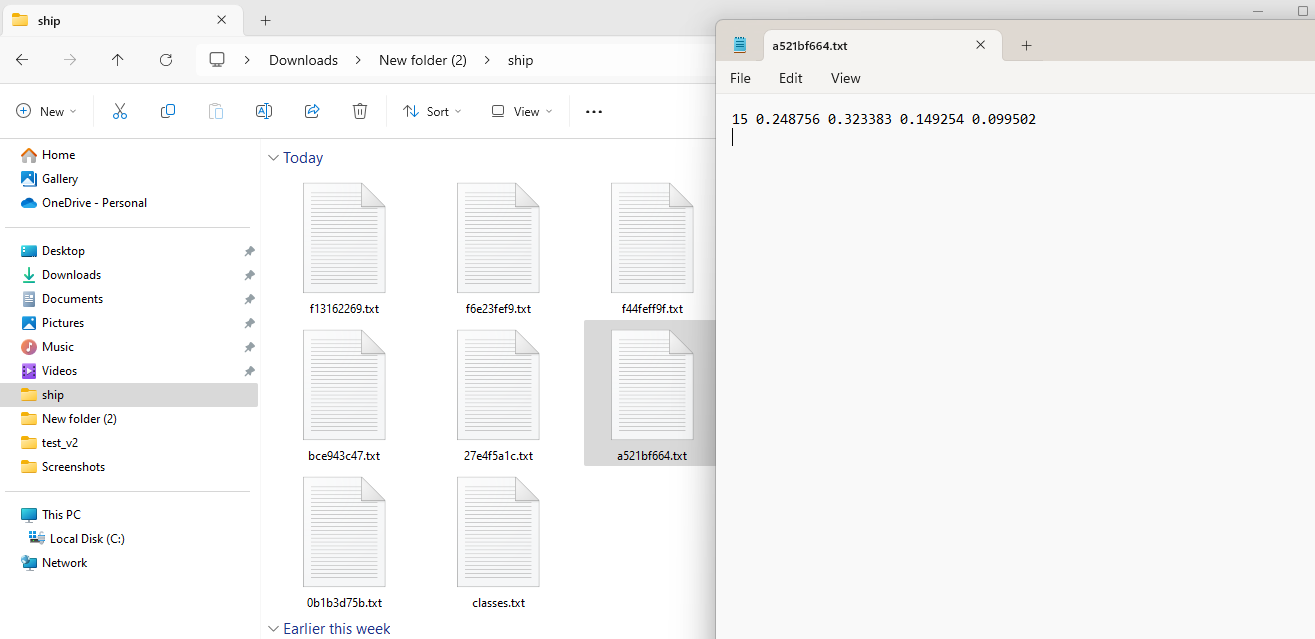


* B3: Chọn yolo và thực hiện gán nhãn bằng create rectbox và điền nhãn cho ảnh và lưu lại:

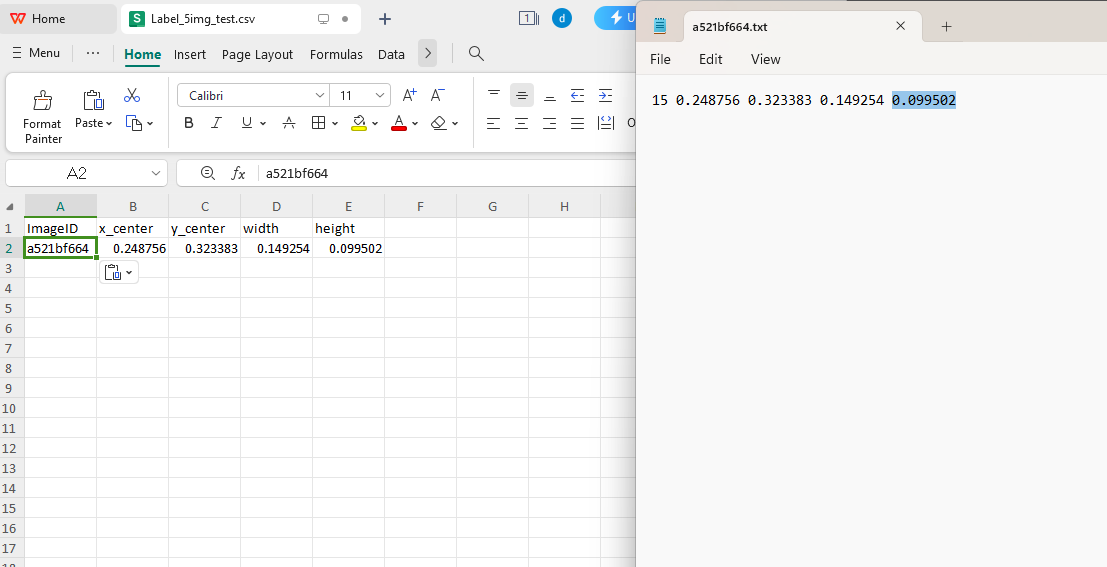




* B4: Sau khi đã lưu ảnh, truy cập vào thư mụ chứa ảnh và mở file .txt của ảnh vừa gán nhãn:



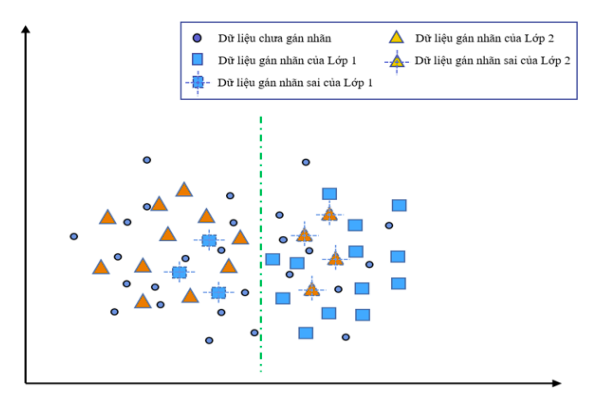
* B5: điền thông số vào csv theo thứ tự và lưu lại:



# II. Phương pháp và các thuật toán bán giám sát mờ

### 1.1. Vấn đề phân cụm dữ liệu với độ tin cậy

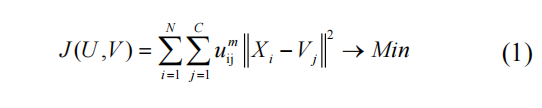
- Ví dụ về phân cụm dữ liệu theo độ tin cậy

****

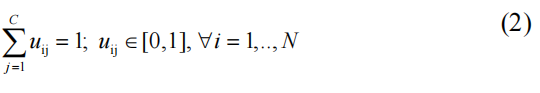
- Tập dữ liệu gồm hai cụm dữ liệu trong đó bao gồm dữ liệu chưa được gán nhãn và có một số dữ liệu đã được gán nhãn (như hình chữ nhật mô phỏng các điểm dữ liệu được gán nhãn của Lớp 1 và các hình tam giác mô phỏng dữ liệu được gán nhãn của Lớp 2). Đường nét đứt ngầm hiển thị ranh giới giữa hai cụm. Trong đó có một số dữ liệu được gán nhãn không chính xác (được biểu diễn bằng dấu thập phía trên các ký hiệu của dữ liệu được gán nhãn). Mục tiêu của bài toán phân cụm dữ liệu với độ tin cậy là tìm ra đường ranh giới “tốt nhất” giữa hai cụm với các dữ liệu được gán nhãn đúng và không chính xác.

### 1.2. Thuật toán phân cụm mờ tiêu chuẩn FCM (Fuzzy C-Mean)

**-** Phương pháp phân cụm mờ tiêu chuẩn FCM (Fuzzy C-Mean) là phương pháp dựa trên cơ sở tối ưu khoảng cách giữa các điểm dữ liệu với trung tâm cụm. Với hàm mục tiêu được xác định như sau:

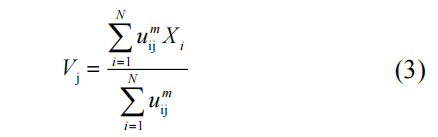


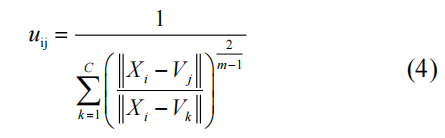
- Với các ràng buộc:



Trong đó tập dữ liệu  với số lượng phần tử ,

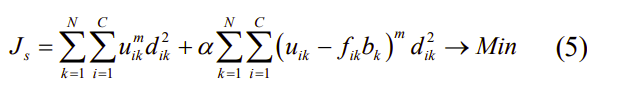
số cụm dữ liệu , tập tâm cụm: , độ thuộc của phần tử với cụm : (uij ), số mờ: (m ). Sử dụng phương pháp Lagrange cho hàm mục tiêu (1) với những ràng buộc (2) thì các trung tâm cụm và các độ thuộc được tính toán như sau:



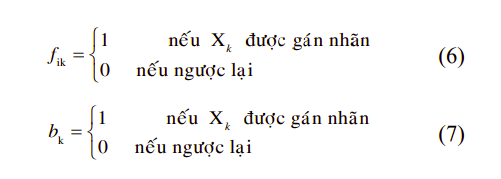


### 1.3. Phân cụm mờ bán giám sát tiêu chuẩn SSFCM (Semi-Supervised Fuzzy C-Means)

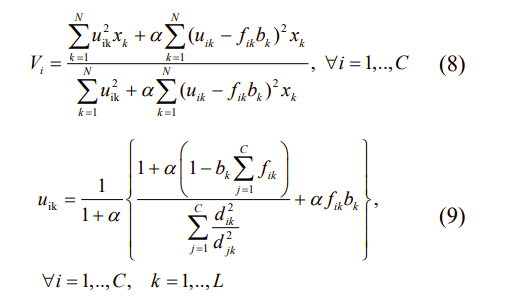
- SSFCM là phương pháp phân cụm bán giám sát được đề xuất bởi Pedrycz and Waletzky [15]. Trong hàm mục tiêu của SSFCM bao gồm 2 thành phần: Thành phần học không giám sát và thành phần học có giám sát. Hàm mục tiêu được xác định như sau:



- Trong đó tập dữ liệu X = X X X X 1 2 , ,..., ,..., k N với số lượng phần tử N , số cụm dữ liệu C , độ thuộc của phần tử k với cụm i : ( ki u ), khoảng cách ki d từ điểm dữ liệu k đến tâm cụm Vi , độ thuộc ki f của điểm dữ liệu được gán nhãn k trong cụm i , số mờ m . Tham số được sử dụng để cân bằng giữa các thành phần học tập được giám sát và không được giám sát là , trong đó k b được sử dụng để phân biệt giữa các phần tử được gán nhãn và không được gán nhãn.



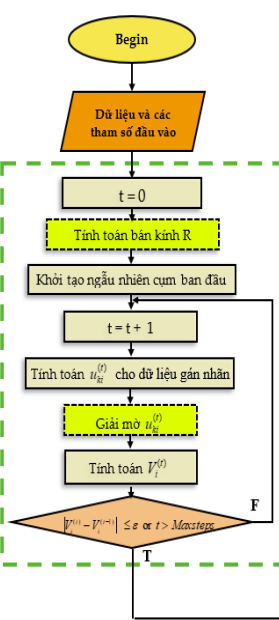
- Các trung tâm cụm Vi và các độ thuộc ki u được tính toán như sau:



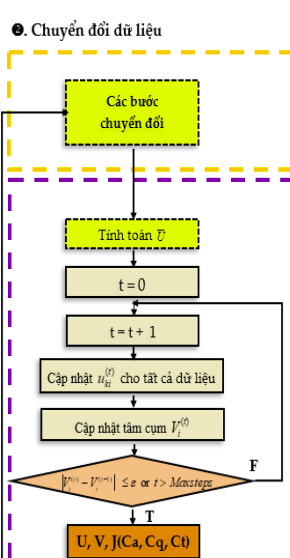
### 1.4. Phương pháp TS3FCM

**-** Thuật toán này có 3 giai đoạn được mô tả bởi 3 khối trong Hình 2 là “FCM cho dữ liệu được gán nhãn”, “Chuyển đổi dữ liệu” và “Phân cụm mờ bán giám sát cho toàn bộ dữ liệu”.

\* Trong khối thứ nhất, dữ liệu được gắn nhãn được thu thập để xác định dữ liệu có độ tin cậy cao và thấp hơn. Để thực hiện được điều này phải sửa đổi thuật toán FCM tiêu chuẩn. Với các đầu vào là dữ liệu và tham số, thuật toán TS3FCM sử dụng FCM trong khối bên trái của Hình 2 để chia dữ liệu được gắn nhãn thành C cụm, trong đó C là số lượng nhãn. Thuật toán cập nhật lặp đi lặp lại các giá trị thành viên U và trung tâm cụm V cho đến khi gặp điều kiện dừng (có nghĩa là sự khác biệt của các trung tâm cụm giữa hai lần lặp liên tiếp nhỏ hơn ngưỡng hoặc số lần lặp lớn hơn ngưỡng tối đa). Trong mỗi lần lặp, các giá trị độ thuộc sẽ được điều chỉnh. Đầu ra của giai đoạn 1 là các trung tâm cụm và các độ thuộc của dữ liệu được gắn nhãn. Lưu ý rằng dữ liệu được gắn nhãn với độ tin cậy cao sẽ có độ thuộc lớn hơn những dữ liệu có độ tin cậy thấp.

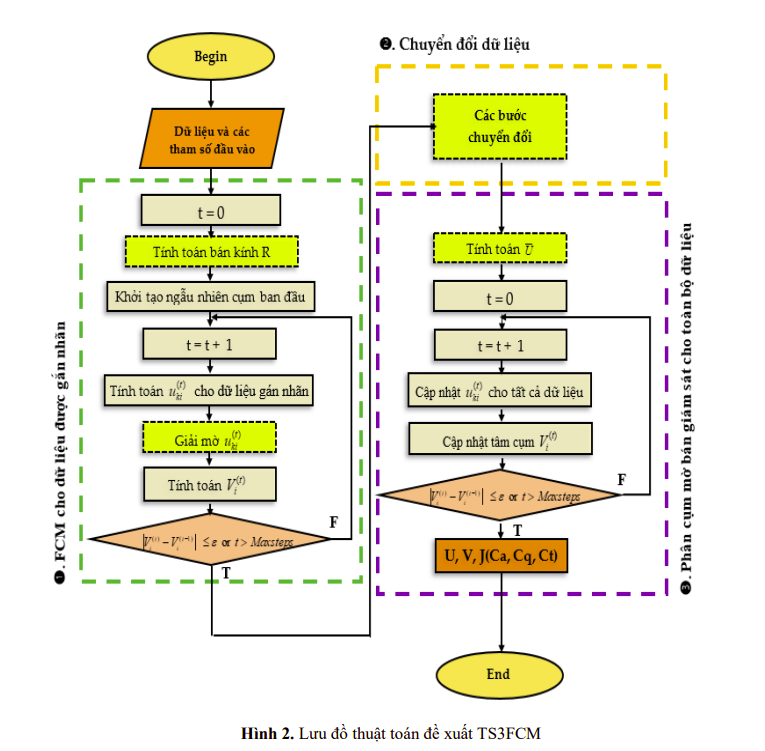
****

**\*** Trong quá trình chuyển kết nối từ khối thứ nhất sang khối thứ hai Chuyển đổi dữ liệu của Hình 2, đầu vào của giai đoạn này là các trung tâm cụm và các độ thuộc của dữ liệu được gắn nhãn của giai đoạn đầu tiên. Các trung tâm cụm của giai đoạn 1 được sử dụng để tính toán các độ thuộc của dữ liệu không được gắn nhãn. Độ thuộc của cả dữ liệu được gắn nhãn và không được gắn nhãn sẽ tạo ra các giá trị độ thuộc đầu vào (Ū). Đầu ra của giai đoạn Chuyển đổi dữ liệu là các độ thuộc của cả dữ liệu được gắn nhãn và không được gắn nhãn, được chứa trong các giá trị độ thuộc đầu vào (Ū).

****

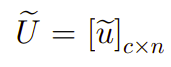
**\*** Trong giai đoạn 3, đầu vào của giai đoạn này là các giá trị độ thuộc ban đầu (Ū) từ giai đoạn Chuyển đổi dữ liệu và các tham số. Các độ thuộc đầu vào (Ū) được sử dụng để hỗ trợ phân cụm mờ bán giám sát để tạo ra các trung tâm cụm cuối cùng và các giá trị thành viên. Đầu ra của giai đoạn 3 là các trung tâm cụm cuối cùng và các giá trị thành viên với các tiêu chí đánh giá về độ chính xác của phân loại, thời gian tính toán và chất lượng phân nhóm như được thể hiện trong khối bên phải của Hình 2.

**Hình 2: Diễn tả các khối này**

****

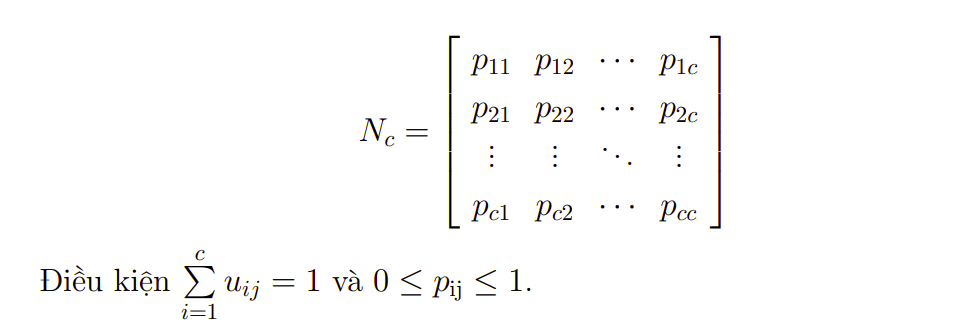
### 1.5. Thuật toán phân cụm bán giám sát mờ an toàn có trọng số tin cậy (CS3FCM)

- Ý tưởng thuật toán: mỗi một phần tử khác nhau thì có một mức độ ảnh hưởng khác nhau đến hiệu suất phân cụm

- Về mặt hình thức, có 2 bộ dữ liệu: tập thứ nhất là X = [x1, x2, . . . , xl ] là bộ dữ liệu được gán nhãn và tập thứ hai Xu = [xl+1, xl+2, . . . xn] là bộ dữ liệu chưa được gán nhãn. Trong đó C là số cụm, phần tử xk có nhãn yk ∈ {1, . . . , c}. Trong phương pháp CS3FCM, các tác giả đã sử dụng FCM để chia tất cả các điểm dữ liệu thành các cụm sau đó tính toán ma trận phân hoạch và ước lượng các nhãn đầu ra  sử dụng thuật toán đối sánh Kuhn–Munkres, nhãn ước   


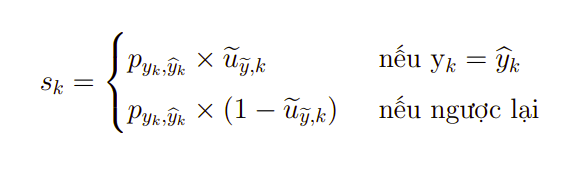


lượng và nhãn chính xác được so sánh để có được ma trận Nc bao gồm các phần tử pij , trong đó pij đo lường khả năng nhãn i được phân vào lớp j.

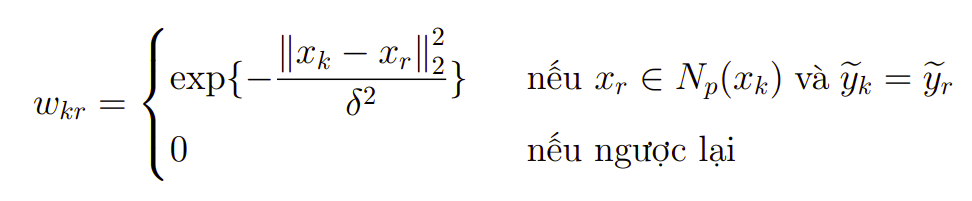




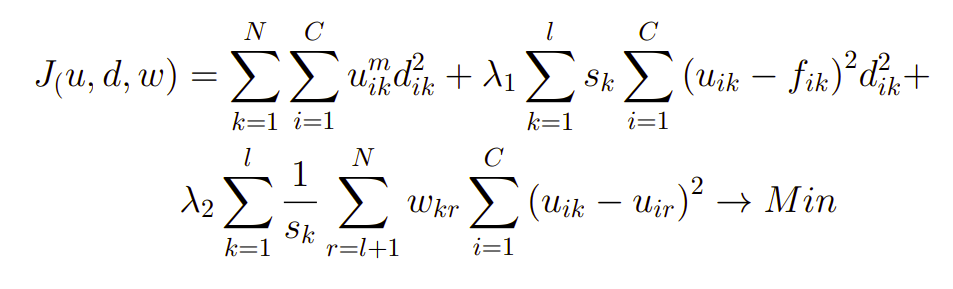
- Đối với một phần tử được gán nhãn xk, nếu yk = và là cao thì độ an toàn tin cậy của nó là cao. Trọng số sk của xk có thể được tính toán như sau:

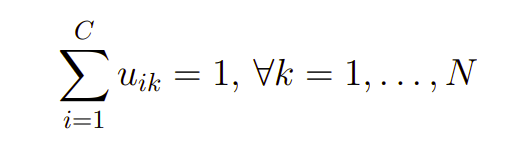


- Thuật toán xây dựng một biểu đồ cục bộ W = [wkr]n×n để định nghĩa các hàng xóm (với các dữ liệu chưa được gán nhãn) cho các phần tử được gán nhãn và xác định trọng số của biểu đồ:

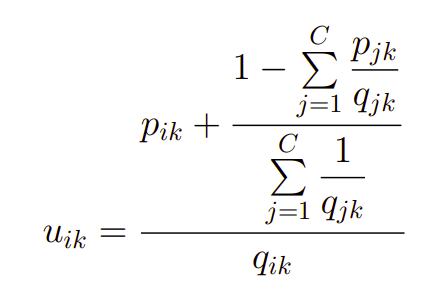


- Trong đó Np(xk) đại diện cho bộ dữ liệu của p hàng xóm gần nhất của các phần tử được gán nhãn xk, trong khi xr đại diện cho các phần tử chưa được gán nhãn. Hàm mục tiêu của CS3FCM được tính như sau:

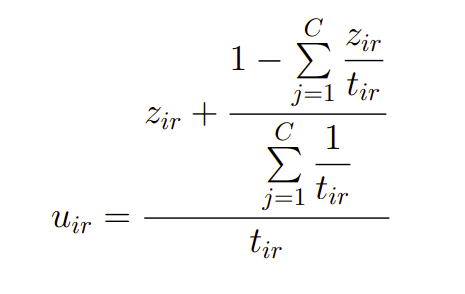


Với ràng buộc:  


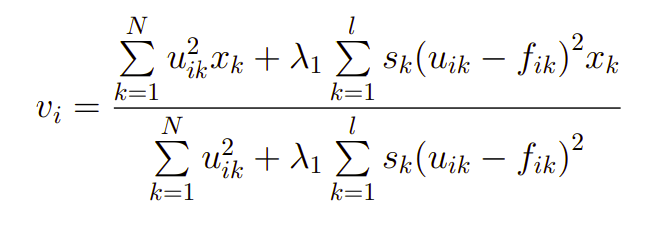
- Độ thuộc uik đối với dữ liệu được gán nhãn xk được tính toán như sau:



- Độ thuộc uir đối với dữ liệu chưa được gán nhãn xr được tính toán bởi công thức sau:



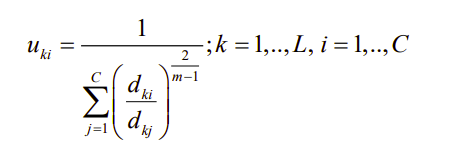
- Tâm cụm Vi được tính bởi:

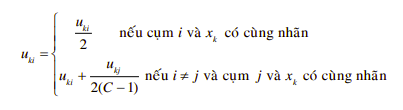


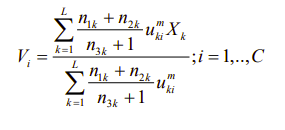
# III. CODE GIẢI THUẬT BÁN GIÁM SÁT MỜ PYTHON

## **A. Thuật toán FCM**

### 1.1. Các bước giải thuật toán FCM

* Thuật toán FCM sửa đổi:
  + Đầu vào: Tập dữ liệu X với số lượng phần tử ( N ), d thuộc tính, số lượng phần tử được gán nhãn trong X : L N ; số lượng cụm C ; ngưỡng ε ; số mờ m ; số mũ ampha và số lần lặp tối đa Maxsteps
  + Đầu ra: Ma trận độ thuộc u và các trung tâm cụm V
* Begin
  + Bước 1: Khởi tạo bước lặp: t = 0
  + Bước 2: Khởi tạo ngẫu nhiên các tâm cụm: Vi (t) <- random i , i = 1,...,C
  + Bước 3: Thực hiện lặp các bước sau đây (4-8):
  + Bước 4: t = t +1
  + Bước 5: Tính toán uki(t)cho dữ liệu được gán nhãn ( k =1 ,...,L ; i=1,...,C ) bởi công thức 
  + Bước 6: Giảm độ thuộc uki(t) theo công thức



* + Bước 7: Tính toán Vi(t) ( i =1,...,C ) theo công thức 
  + Bước 8: Kiểm tra điều kiện dừng của thuật toán theo công thức



nếu thoả mãn thì dừng thuật toán.

### 1.2. Code thuật toán FCM

* **class FuzzyCMeans**:

def \_\_init\_\_(self, n\_clusters=2, m=2, max\_iter=100, tol=1e-4, check=0):

self.n\_clusters = n\_clusters

self.m = m

self.max\_iter = max\_iter

self.tol = tol

self.check = check

self.centers = None

self.U = None

def initialize\_centers(self, X):

random\_indices = np.random.choice(X.shape[0], self.n\_clusters, replace=False)

centers = X[random\_indices]

return centers

def initialize\_centers\_kmeans\_plusplus(self, X):

n\_samples, n\_features = X.shape

centers = np.empty((self.n\_clusters, n\_features))

# Choose the first center randomly

centers[0] = X[np.random.choice(n\_samples)]

# Compute remaining centers

for c\_id in range(1, self.n\_clusters):

distances = np.min([np.sum((X - center) \*\* 2, axis=1) for center in centers[:c\_id]], axis=0)

probabilities = distances / np.sum(distances)

cumulative\_probabilities = np.cumsum(probabilities)

r = np.random.rand()

next\_center = X[np.searchsorted(cumulative\_probabilities, r)]

centers[c\_id] = next\_center

return centers

def initialize\_membership\_matrix(self, n\_samples):

U = np.random.dirichlet(np.ones(self.n\_clusters), size=n\_samples)

return U

def update\_centers(self, X):

if self.check == 0:

um = self.U \*\* self.m

centers = (um.T @ X) / um.sum(axis=0)[:, None]

elif self.check == 1:

n\_labeled = self.labeled\_data.shape[0]

n\_unlabeled = X.shape[0] - n\_labeled

um = self.U \*\* self.m

centers = np.zeros((self.n\_clusters, X.shape[1]))

for i in range(self.n\_clusters):

numerator = np.sum([(n\_labeled \* (1 + 1/n\_labeled) + n\_unlabeled) \* um[k, i] \* X[k] for k in range(X.shape[0])], axis=0)

denominator = (n\_labeled \* (1 + 1/n\_labeled) + n\_unlabeled) \* um[:, i].sum()

centers[i] = numerator / denominator

return centers

def update\_membership\_matrix(self, X):

n\_samples = X.shape[0]

new\_U = np.zeros((n\_samples, self.n\_clusters))

if self.check == 0:

for i in range(n\_samples):

for j in range(self.n\_clusters):

denominator = sum((np.linalg.norm(X[i] - self.centers[j]) / np.linalg.norm(X[i] - self.centers[k])) \*\* (2 / (self.m - 1)) for k in range(self.n\_clusters))

new\_U[i, j] = 1 / denominator

elif self.check == 1:

n\_labeled = self.labeled\_data.shape[0]

for i in range(n\_samples):

for j in range(self.n\_clusters):

if i < n\_labeled:

if self.labels[i] == j:

new\_U[i, j] = 1

else:

new\_U[i, j] = 1e-10 # giá trị nhỏ để tránh chia cho 0

else:

denominator = sum((np.linalg.norm(X[i] - self.centers[j]) / np.linalg.norm(X[i] - self.centers[k])) \*\* (2 / (self.m - 1)) for k in range(self.n\_clusters))

new\_U[i, j] = 1 / denominator

return new\_U

def objective\_function(self, X):

if self.check == 0:

um = self.U \*\* self.m

obj = np.sum(um \* np.linalg.norm(X[:, np.newaxis] - self.centers, axis=2) \*\* 2)

elif self.check == 1:

n\_labeled = self.labeled\_data.shape[0]

n\_unlabeled = X.shape[0] - n\_labeled

um = self.U \*\* self.m

obj = np.sum((n\_labeled + n\_unlabeled) \* um \* np.linalg.norm(X[:, np.newaxis] - self.centers, axis=2) \*\* 2)

return obj

def fit(self, X):

n\_samples = X.shape[0]

self.U = self.initialize\_membership\_matrix(n\_samples)

self.centers = self.initialize\_centers\_kmeans\_plusplus(X)

for iteration in range(self.max\_iter):

self.centers = self.update\_centers(X)

new\_U = self.update\_membership\_matrix(X)

objective\_value = self.objective\_function(X)

if self.check == 0:

print(f'FCM: {iteration + 1}, Objective function: {objective\_value}')

elif self.check == 1:

print(f'FCM improve: {iteration + 1}, Objective function: {objective\_value}')

if np.linalg.norm(new\_U - self.U) < self.tol:

print(f'Converged after {iteration + 1} iterations')

break

self.U = new\_U

def predict(self, X):

new\_U = self.update\_membership\_matrix(X)

return np.argmax(new\_U, axis=1)

### 1.3. Code Thuật toán FCM cải tiến

* **class FuzzyCMeansImprove:**

def \_\_init\_\_(self, n\_clusters=2, m=2, max\_iter=100, tol=1e-4):

self.n\_clusters = n\_clusters

self.m = m

self.max\_iter = max\_iter

self.tol = tol

self.centers = None

self.U = None

def initialize\_centers(self, X):

random\_indices = np.random.choice(X.shape[0], self.n\_clusters, replace=False)

centers = X[random\_indices]

return centers

def calculate\_distance\_matrix(self, X, centers):

distance\_matrix = np.zeros((X.shape[0], self.n\_clusters))

for j in range(self.n\_clusters):

distances = np.linalg.norm(X - centers[j], axis=1)

distance\_matrix[:, j] = distances

return distance\_matrix

def update\_membership\_matrix(self, X, distance\_matrix):

new\_U = np.zeros((X.shape[0], self.n\_clusters))

for i in range(X.shape[0]):

for j in range(self.n\_clusters):

if np.any(distance\_matrix[i, :] == 0):

new\_U[i, j] = 0

else:

denominator = np.sum((distance\_matrix[i, j] / distance\_matrix[i, :]) \*\* (2 / (self.m - 1)))

if np.isnan(denominator):

new\_U[i, j] = 0

else:

new\_U[i, j] = 1 / denominator

return new\_U

def update\_membership\_strength(self, U):

new\_U = np.copy(U)

for i in range(U.shape[0]):

for j in range(self.n\_clusters):

new\_U[i, j] /= 2

return new\_U

def update\_centers(self, X, U):

numerator = np.dot(U.T \*\* self.m, X)

denominator = np.sum(U \*\* self.m, axis=0).reshape(-1, 1)

centers = numerator / denominator

return centers

def check\_convergence(self, centers, old\_centers):

if np.linalg.norm(centers - old\_centers) <= self.tol:

return True

else:

return False

def objective\_function(self, X):

um = self.U \*\* self.m

obj = np.sum(um \* np.linalg.norm(X[:, np.newaxis] - self.centers, axis=2) \*\* 2)

return obj

def fit(self, X):

self.centers = self.initialize\_centers(X)

t = 0

while t < self.max\_iter:

old\_centers = np.copy(self.centers)

distance\_matrix = self.calculate\_distance\_matrix(X, self.centers)

self.U = self.update\_membership\_matrix(X, distance\_matrix)

self.U = self.update\_membership\_strength(self.U)

self.centers = self.update\_centers(X, self.U)

objective\_value = self.objective\_function(X)

print(f'FCM Improve: {t + 1}, Objective function: {objective\_value}')

if self.check\_convergence(self.centers, old\_centers):

print(f'Converged after {t + 1} iterations.')

break

new\_U = self.update\_membership\_matrix(X, self.calculate\_distance\_matrix(X, self.centers))

if np.linalg.norm(new\_U - self.U) < self.tol:

print(f'Converged after {t + 1} iterations')

break

self.U = new\_U

t += 1

def predict(self, X):

distance\_matrix = self.calculate\_distance\_matrix(X, self.centers)

new\_U = self.update\_membership\_matrix(X, distance\_matrix)

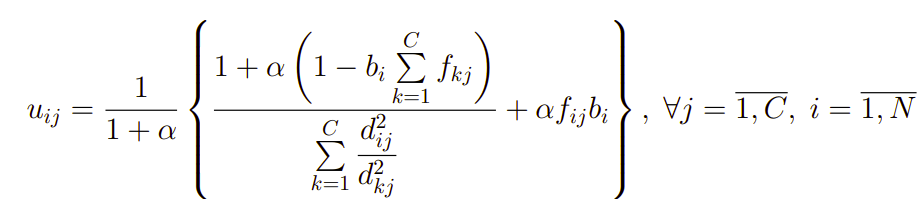
predicted\_labels = np.argmax(new\_U, axis=1)

return predicted\_labels

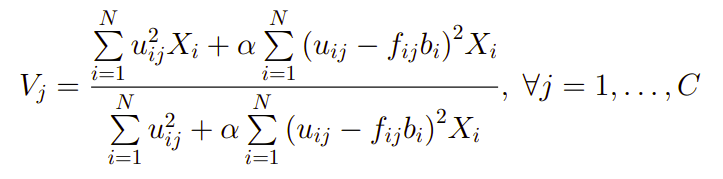
## **B. Thuật toán SSFCM**

### 2.1. Các bước giải thuật toán SSFCM

* Thuật toán SSFCM:
  + Đầu vào: Bộ dữ liệu X = {X1, X2, . . . , Xk, . . . , XN } với số điểm dữ liệu N; số cụm C; độ thuộc của điểm dữ liệu i đối với cụm j : uij , tham số mờ m; ngưỡng ε và số lần lặp tối đa MaxStep > 0.
  + Đầu ra: Ma trận độ thuộc u và tâm cụm V .
* Begin
  + Bước 1: t = 0
  + Bước 2: Khởi tạo ngẫu nhiên tâm cụm ban đầu: V (0) ← random
  + Bước 3: Thực hiện lặp các bước từ 4-7:
  + Bước 4: t = t + 1
  + Bước 5: Tính toán uij(t) bởi công thức



* + Bước 6: Tính toán Vj(t) bởi công thức



* + Bước 7: Kiểm tra điều kiện dừng: || Vj(t) - Vj(t-1)|| ≤ ε hoặc t > MaxStep. Nếu thoả mãn điều kiện thì dừng thuật toán, nếu không thoả mãn thì quay lại Bước 3

### 2.2. Code thuật toán SSFCM

* **class SSFCM:**

def \_\_init\_\_(self, n\_clusters=2, m=2, alpha=0.5, max\_iter=100, tol=1e-4, check=0):

self.n\_clusters = n\_clusters

self.m = m

self.alpha = alpha

self.max\_iter = max\_iter

self.tol = tol

self.check = check

self.centers = None

self.U = None

self.f = None

self.b = None

def initialize\_centers(self, X):

random\_indices = np.random.choice(X.shape[0], self.n\_clusters, replace=False)

centers = X[random\_indices]

return centers

def initialize\_membership\_matrix(self, n\_samples):

U = np.random.dirichlet(np.ones(self.n\_clusters), size=n\_samples)

return U

def update\_centers(self, X):

um = self.U \*\* self.m

um\_alpha = (self.U - self.U \* self.b[:, np.newaxis]) \*\* self.m

num = np.dot(um.T, X) + self.alpha \* np.dot(um\_alpha.T, X)

den = um.sum(axis=0)[:, None] + self.alpha \* um\_alpha.sum(axis=0)[:, None]

centers = num / den

return centers

def update\_membership\_matrix(self, X):

n\_samples = X.shape[0]

new\_U = np.zeros((n\_samples, self.n\_clusters))

distances = np.zeros((n\_samples, self.n\_clusters))

for i in range(self.n\_clusters):

distances[:, i] = np.linalg.norm(X - self.centers[i], axis=1)

for k in range(n\_samples):

for i in range(self.n\_clusters):

numerator = 1 + self.alpha \* (1 - self.b[k] \* np.sum(self.f[k, :]))

denominator = np.sum([(distances[k, i] / distances[k, j]) \*\* (2 / (self.m - 1)) for j in range(self.n\_clusters)])

new\_U[k, i] = (numerator / denominator) / (1 + self.alpha) + self.alpha \* self.f[k, i] \* self.b[k]

return new\_U

def objective\_function(self, X):

n\_samples = X.shape[0]

um = self.U \*\* self.m

d = np.zeros((n\_samples, self.n\_clusters))

for i in range(self.n\_clusters):

d[:, i] = np.linalg.norm(X - self.centers[i], axis=1)

term1 = np.sum(um \* (d \*\* 2))

um\_alpha = (self.U - self.U \* self.b[:, np.newaxis]) \*\* self.m

term2 = self.alpha \* np.sum(um\_alpha \* (d \*\* 2))

return term1 + term2

def fit(self, X, labeled\_pixel\_index):

n\_samples = X.shape[0]

self.U = self.initialize\_membership\_matrix(n\_samples)

self.centers = self.initialize\_centers(X)

self.f = np.zeros((n\_samples, self.n\_clusters))

self.b = np.zeros(n\_samples)

self.f[labeled\_pixel\_index, 0] = 1

self.b[labeled\_pixel\_index] = 1

for iteration in range(self.max\_iter):

self.centers = self.update\_centers(X)

new\_U = self.update\_membership\_matrix(X)

# Tính toán hàm mục tiêu

objective\_value = self.objective\_function(X)

if self.check == 0:

print(f'SSFCM: {iteration + 1}, Objective function: {objective\_value}')

elif self.check == 1:

print(f'CS3FCM: {iteration + 1}, Objective function: {objective\_value}')

elif self.check == 2:

print(f'TS3FCM: {iteration + 1}, Objective function: {objective\_value}')

# Kiểm tra điều kiện hội tụ

if np.linalg.norm(new\_U - self.U) < self.tol:

print(f'Converged after {iteration + 1} iterations')

break

self.U = new\_U

def predict(self, X):

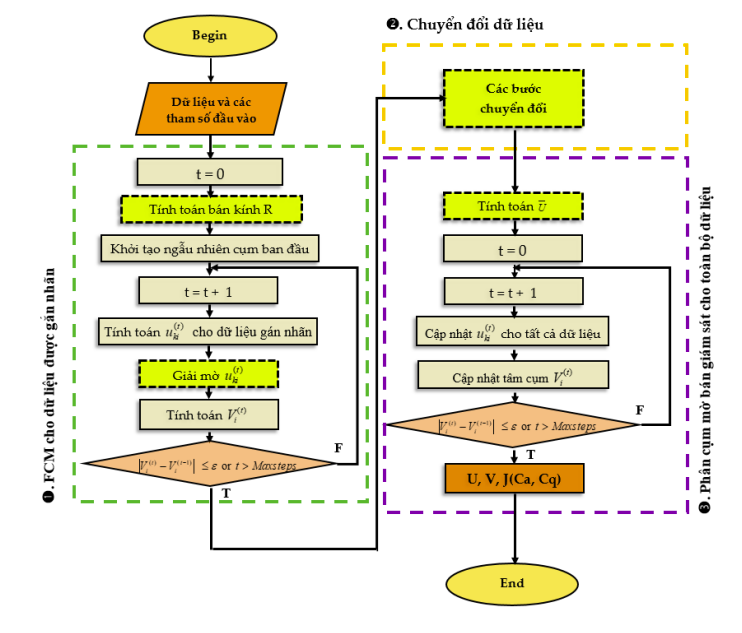
new\_U = self.update\_membership\_matrix(X)

return np.argmax(new\_U, axis=1)

## **C. Thuật toán TS3FCM**

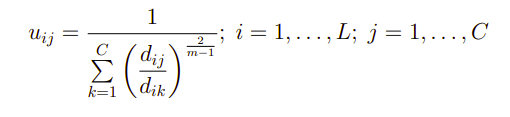
### 3.1. Các bước giải thuật toán TS3FCM

- Chi tiết thuật toán TS3FCM:

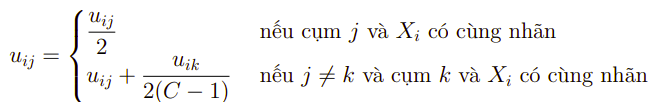


\* Bước 1:Phân cụm mờ cho dữ liệu đã được gán nhãn

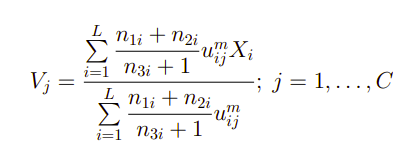
* Thuật toán FCM cải tiến:
  + Đầu vào: Bộ dữ liệu X = {X1, X2, . . . , Xk, . . . , XN } với số điểm dữ liệu N; số điểm dữ liệu được gán nhãn trong X: L < N; số cụm C; ngưỡng ε; số mờ m; số mũ α và số lần lặp tối đa MaxStep > 0. Đầu ra: Ma trận độ thuộc u và tâm cụm V
  + Đầu ra: Ma trận độ thuộc u và tâm cụm V .
* Begin
  + Bước 1: t = 0
  + Bước 2: Khởi tạo ngẫu nhiên các tâm cụm: Vi(t) <- random, j = 1,...,C
  + Bước 3: Thực hiện lặp các bước sau đây (4-8):
  + Bước 4: t = t +1
  + Bước 5: Tính toán uij(t)cho dữ liệu được gán nhãn ( i =1 ,...,L ; j=1,...,C ) bởi công thức



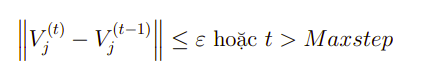
* + Bước 6: Giảm độ thuộc uij(t) theo công thức



* + Bước 7: Tính toán Vj(t) ( j = 1, C) theo công thức



* + Bước 8: Kiểm tra điều kiện dừng của thuật toán theo công thức



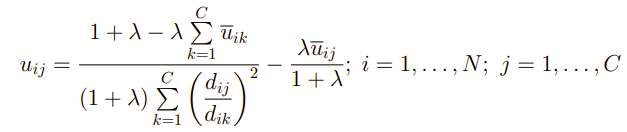
nếu thoả mãn thì dừng thuật toán, nếu không thỏa mãn thì quay lại Bước 3.

\* Bước 2: Chuyển đổi dữ liệu

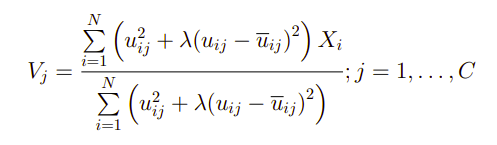
* Từ kết quả đầu ra của khối thứ nhất sẽ thu được các tâm cụm của dữ liệu được gán nhãn và sử dụng chúng làm tâm cụm ban đầu cho bộ dữ liệu của các điểm dữ liệu chưa được gán nhãn. Ở đây, sử dụng thuật toán FCM cho dữ liệu chưa được gán nhãn để thu được các giá trị độ thuộc của dữ liệu chưa được gán nhãn. Sự kết hợp các giá trị độ thuộc của cả dữ liệu được gán nhãn và chưa được gán nhãn tạo nên các giá trị độ thuộc đầu vào (U) cho tất cả các điểm dữ liệu của bước tiếp theo.

\* Bước 3: Phân cụm bán giám sát mờ cho toàn bộ dữ liệu

* Thuật toán phân cụm bán giám sát mờ mới
  + Đầu vào: Bộ dữ liệu X = {X1, X2, . . . , Xk, . . . , XN } với số điểm dữ liệu N; số điểm dữ liệu được gán nhãn trong X: L < N; số cụm C; ngưỡng ε; số mờ m; số mũ α và số lần lặp tối đa MaxStep > 0. Đầu ra: Ma trận độ thuộc u và tâm cụm V
  + Đầu ra: Ma trận độ thuộc u và tâm cụm V .
* Begin
  + Bước 1: t = 0
  + Bước 2: Thực hiện lặp các bước sau đây (3-6)
  + Bước 3: t = t + 1
  + Bước 4: Tính toán uij(t) (i = 1, . . . , N, j = 1, . . . , C) bởi công thức



* + Bước 5: Tính toán Vj(t) (j = 1, . . . , C) theo công thức



* + Kiểm tra điều kiện dừng: || Vj(t) - Vj(t-1)|| ≤ ε hoặc t > MaxStep.. Nếu thoả mãn điều kiện thì dừng thuật toán, nếu không thỏa mãn thì quay lại Bước 2

### 3.2. Code thuật toán TS3FCM

* Thực hiện trong main:

def create\_labeled\_mask\_fcm(image\_path, box):

image = Image.open(image\_path)

image\_width, image\_height = image.size

pixel\_box = convert\_normalized\_box\_to\_pixel(box, image\_width, image\_height)

x\_min, y\_min, x\_max, y\_max = pixel\_box

cropped\_image = np.array(image)[y\_min:y\_max, x\_min:x\_max]

cropped\_image\_gray = cv2.cvtColor(cropped\_image, cv2.COLOR\_BGR2GRAY)

fcm\_improve = FuzzyCMeansImprove(n\_clusters=2, m=2, max\_iter=100, tol=1e-4)

fcm\_improve.fit(cropped\_image\_gray.reshape(-1, 1))

labels\_improve = fcm\_improve.predict(cropped\_image\_gray.reshape(-1, 1)).reshape(cropped\_image\_gray.shape)

mask\_improve = (labels\_improve == 1).astype(np.uint8) # Giả sử nhãn '1' là tàu

full\_mask\_improve = np.zeros((image\_height, image\_width), dtype=np.uint8)

full\_mask\_improve[y\_min:y\_max, x\_min:x\_max] = mask\_improve

return full\_mask\_improve

def segment\_image\_with\_box(image\_path, box, n\_clusters=2):

image = io.imread(image\_path)

image\_lab = color.rgb2lab(image)

image\_height, image\_width = image.shape[:2]

pixel\_box = convert\_normalized\_box\_to\_pixel(box, image\_width, image\_height)

labeled\_mask = create\_mask\_from\_box(image\_height, image\_width, pixel\_box)

labeled\_pixel\_index = np.where(labeled\_mask.flatten() == 1)[0]

image\_data = image\_lab.reshape((-1, 3)).astype(np.float32) / 255.0

labeled\_mask\_fcm\_improve = create\_labeled\_mask\_fcm(image\_path, box)

ts3fcm = SSFCM(n\_clusters=n\_clusters, m=2, alpha=0.5, max\_iter=100, tol=0.0001, check=2)

ts3fcm.fit(image\_data, labeled\_mask\_fcm\_improve)

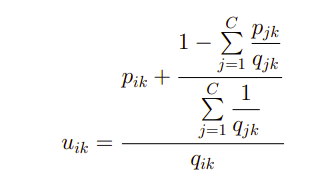
segmented\_image\_ts3fcm = np.argmax(ts3fcm.U, axis=1).reshape(image\_lab.shape[:2])

return image, labeled\_mask, labeled\_mask\_fcm\_improve, segmented\_image\_ts3fcm

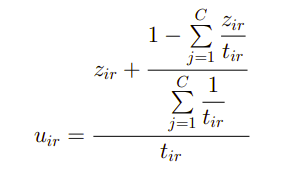
## **D. Thuật toán CS3FCM**

### 4.1. Các bước giải thuật toán CS3FCM

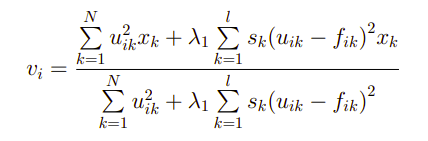
* Thuật toán CS3FCM:
  + Đầu vào: Bộ dữ liệu X = {X1, X2, . . . , Xk, . . . , XN } với số điểm dữ liệu N; số cụm C; độ thuộc của điểm dữ liệu i đối với cụm j : uij , tham số mờ m; ngưỡng ε và số lần lặp tối đa MaxStep > 0.
  + Đầu ra: Ma trận độ thuộc u và tâm cụm V .
* Begin
  + Bước 1: Thực hiện phân cụm FCM trên bộ dữ liệu Xl ∪ Xu để thu được kết quả ước lượng nhãn đầu ra Y˜ , ma trận phân hoạch U˜ và ma trận NC
  + Bước 2: Tính toán giá trị trọng số sk
  + Bước 3: Xây dựng biểu đồ cục bộ W
  + Bước 4: t = 0
  + Bước 5: Khởi tạo ngẫu nhiên các tâm cụm: Vi(t) <- random, j = 1,...,C
  + Bước 6: Thực hiện lặp các bước từ 7-12:
  + Bước 7: t = t + 1
  + Bước 8: Tính toán uik(t) bởi công thức



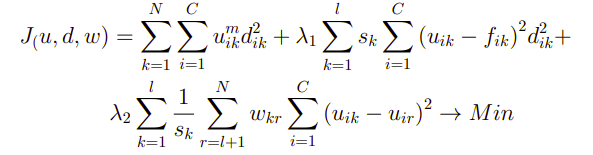
* + Bước 9: Tính toán uir(t) bởi công thức



* + Bước 10: Tính toán vi(t) bởi công thức



* + Bước 11: Tính toán giá trị của J(t) bởi công thức



* + Bước 12: Kiểm tra điều kiện dừng: | J (t) − J (t−1)| ≤ η hoặc t > M axStep. Nếu thoả mãn điều kiện thì dừng thuật toán, nếu không thoả mãn thì quay lại Bước 6.

### 4.2. Code thuật toán CS3FCM

* Thực hiện trong main

def create\_labeled\_mask\_fcm(image\_path, box):

image = Image.open(image\_path)

image\_width, image\_height = image.size

pixel\_box = convert\_normalized\_box\_to\_pixel(box, image\_width, image\_height)

x\_min, y\_min, x\_max, y\_max = pixel\_box

cropped\_image = np.array(image)[y\_min:y\_max, x\_min:x\_max]

cropped\_image\_gray = cv2.cvtColor(cropped\_image, cv2.COLOR\_BGR2GRAY)

fcm = FuzzyCMeans(n\_clusters=2, m=2, max\_iter=100, tol=1e-4)

fcm.fit(cropped\_image\_gray.reshape(-1, 1))

labels = fcm.predict(cropped\_image\_gray.reshape(-1, 1)).reshape(cropped\_image\_gray.shape)

mask = (labels == 1).astype(np.uint8) # Giả sử nhãn '1' là tàu

full\_mask = np.zeros((image\_height, image\_width), dtype=np.uint8)

full\_mask[y\_min:y\_max, x\_min:x\_max] = mask

return full\_mask

def segment\_image\_with\_box(image\_path, box, n\_clusters=2):

image = io.imread(image\_path)

image\_lab = color.rgb2lab(image)

image\_height, image\_width = image.shape[:2]

pixel\_box = convert\_normalized\_box\_to\_pixel(box, image\_width, image\_height)

labeled\_mask = create\_mask\_from\_box(image\_height, image\_width, pixel\_box)

labeled\_pixel\_index = np.where(labeled\_mask.flatten() == 1)[0]

image\_data = image\_lab.reshape((-1, 3)).astype(np.float32) / 255.0

labeled\_mask\_fcm = create\_labeled\_mask\_fcm(image\_path, box)

cs3fcm = SSFCM(n\_clusters=n\_clusters, m=2, alpha=0.5, max\_iter=100, tol=0.0001, check=1)

cs3fcm.fit(image\_data, labeled\_mask\_fcm)

segmented\_image\_cs3fcm = np.argmax(cs3fcm.U, axis=1).reshape(image\_lab.shape[:2])

return image, labeled\_mask, labeled\_mask\_fcm, segmented\_image\_cs3fcm

## **E. Code Tổng hợp**

### 5.1. Thư viện cần dùng

* **Main:**
* import numpy as np
* import cv2
* from skimage import io, color
* from PIL import Image
* import matplotlib.pyplot as plt
* import pandas as pd
* import os
* from fcm import FuzzyCMeans
* from ssfcm import SSFCM
* from fcm\_improve import FuzzyCMeansImprove
* **Gui:**
* import os
* import sys
* import tkinter as tk
* from tkinter import filedialog
* from PIL import Image
* import pandas as pd
* import numpy as np
* import cv2
* import time
* from fcm import FuzzyCMeans
* from ssfcm import SSFCM
* from fcm\_improve import FuzzyCMeansImprove
* from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg
* from matplotlib.figure import Figure
* from skimage import io, color
* from tkinter import messagebox

### 5.2 Main

* **def convert\_normalized\_box\_to\_pixel(box, image\_width, image\_height):**

x\_center, y\_center, width, height = box

x\_min = int((x\_center - width / 2) \* image\_width)

y\_min = int((y\_center - height / 2) \* image\_height)

x\_max = int((x\_center + width / 2) \* image\_width)

y\_max = int((y\_center + height / 2) \* image\_height)

return x\_min, y\_min, x\_max, y\_max

* **def create\_mask\_from\_box(image\_height, image\_width, box):**

mask = np.zeros((image\_height, image\_width), dtype=np.uint8)

x\_min, y\_min, x\_max, y\_max = box

mask[y\_min:y\_max, x\_min:x\_max] = 1

return mask

* **def create\_labeled\_mask\_fcm(image\_path, box):**

image = Image.open(image\_path)

image\_width, image\_height = image.size

pixel\_box = convert\_normalized\_box\_to\_pixel(box, image\_width, image\_height)

x\_min, y\_min, x\_max, y\_max = pixel\_box

cropped\_image = np.array(image)[y\_min:y\_max, x\_min:x\_max]

cropped\_image\_gray = cv2.cvtColor(cropped\_image, cv2.COLOR\_BGR2GRAY)

fcm = FuzzyCMeans(n\_clusters=2, m=2, max\_iter=100, tol=1e-4)

fcm.fit(cropped\_image\_gray.reshape(-1, 1))

fcm\_improve = FuzzyCMeansImprove(n\_clusters=2, m=2, max\_iter=100, tol=1e-4)

fcm\_improve.fit(cropped\_image\_gray.reshape(-1, 1))

labels = fcm.predict(cropped\_image\_gray.reshape(-1, 1)).reshape(cropped\_image\_gray.shape)

mask = (labels == 1).astype(np.uint8) # Giả sử nhãn '1' là tàu

labels\_improve = fcm\_improve.predict(cropped\_image\_gray.reshape(-1, 1)).reshape(cropped\_image\_gray.shape)

mask\_improve = (labels\_improve == 1).astype(np.uint8) # Giả sử nhãn '1' là tàu

full\_mask = np.zeros((image\_height, image\_width), dtype=np.uint8)

full\_mask[y\_min:y\_max, x\_min:x\_max] = mask

full\_mask\_improve = np.zeros((image\_height, image\_width), dtype=np.uint8)

full\_mask\_improve[y\_min:y\_max, x\_min:x\_max] = mask\_improve

return full\_mask, full\_mask\_improve

* **def segment\_image\_with\_box(image\_path, box, n\_clusters=2):**

image = io.imread(image\_path)

image\_lab = color.rgb2lab(image)

image\_height, image\_width = image.shape[:2]

pixel\_box = convert\_normalized\_box\_to\_pixel(box, image\_width, image\_height)

labeled\_mask = create\_mask\_from\_box(image\_height, image\_width, pixel\_box)

labeled\_pixel\_index = np.where(labeled\_mask.flatten() == 1)[0]

image\_data = image\_lab.reshape((-1, 3)).astype(np.float32) / 255.0

labeled\_mask\_fcm, labeled\_mask\_fcm\_improve = create\_labeled\_mask\_fcm(image\_path, box)

sscfm = SSFCM(n\_clusters=n\_clusters, m=2, alpha=0.5, max\_iter=100, tol=0.0001, check=0)

sscfm.fit(image\_data, labeled\_pixel\_index)

cs3fcm = SSFCM(n\_clusters=n\_clusters, m=2, alpha=0.5, max\_iter=100, tol=0.0001, check=1)

cs3fcm.fit(image\_data, labeled\_mask\_fcm)

ts3fcm = SSFCM(n\_clusters=n\_clusters, m=2, alpha=0.5, max\_iter=100, tol=0.0001, check=2)

ts3fcm.fit(image\_data, labeled\_mask\_fcm\_improve)

segmented\_image\_ssfcm = np.argmax(sscfm.U, axis=1).reshape(image\_lab.shape[:2])

segmented\_image\_cs3fcm = np.argmax(cs3fcm.U, axis=1).reshape(image\_lab.shape[:2])

segmented\_image\_ts3fcm = np.argmax(ts3fcm.U, axis=1).reshape(image\_lab.shape[:2])

return image, labeled\_mask, labeled\_mask\_fcm, labeled\_mask\_fcm\_improve, segmented\_image\_ssfcm, segmented\_image\_cs3fcm, segmented\_image\_ts3fcm

* **def display\_segmented\_image(image, labeled\_mask, labeled\_mask\_fcm, labeled\_mask\_fcm\_improve, segmented\_image\_ssfcm, segmented\_image\_cs3fcm, segmented\_image\_ts3fcm):**

contour\_image\_ssfcm = image.copy()

contour\_image\_cs3fcm = image.copy()

contour\_image\_ts3fcm = image.copy()

for label in np.unique(segmented\_image\_ssfcm):

mask = (segmented\_image\_ssfcm == label).astype(np.uint8)

contours, \_ = cv2.findContours(mask, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

cv2.drawContours(contour\_image\_ssfcm, contours, -1, (255, 0, 0), 1)

for label in np.unique(segmented\_image\_cs3fcm):

mask = (segmented\_image\_cs3fcm == label).astype(np.uint8)

contours, \_ = cv2.findContours(mask, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

cv2.drawContours(contour\_image\_cs3fcm, contours, -1, (255, 0, 0), 1)

for label in np.unique(segmented\_image\_ts3fcm):

mask = (segmented\_image\_ts3fcm == label).astype(np.uint8)

contours, \_ = cv2.findContours(mask, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

cv2.drawContours(contour\_image\_ts3fcm, contours, -1, (255, 0, 0), 1)

plt.figure(figsize=(15, 7))

plt.subplot(3, 3, 1)

plt.imshow(image)

plt.title('Original Image')

plt.axis('off')

plt.subplot(3, 3, 2)

plt.imshow(labeled\_mask, cmap='gray')

plt.title('Labeled Mask')

plt.axis('off')

plt.subplot(3, 3, 4)

plt.imshow(labeled\_mask\_fcm, cmap='gray')

plt.title('Labeled Mask FCM')

plt.axis('off')

plt.subplot(3, 3, 5)

plt.imshow(labeled\_mask\_fcm\_improve, cmap='gray')

plt.title('Labeled Mask FCM Improve')

plt.axis('off')

plt.subplot(3, 3, 7)

plt.imshow(contour\_image\_ssfcm)

plt.title('SSFCM')

plt.axis('off')

plt.subplot(3, 3, 8)

plt.imshow(contour\_image\_cs3fcm)

plt.title('CS3FCM')

plt.axis('off')

plt.subplot(3, 3, 9)

plt.imshow(contour\_image\_ts3fcm)

plt.title('TS3FCM')

plt.axis('off')

plt.tight\_layout()

plt.show()

csv\_path = 'Label\_img.csv'

df = pd.read\_csv(csv\_path)

n\_clusters = 2

image\_folder = 'ship'

for image\_name in os.listdir(image\_folder):

image\_path = os.path.join(image\_folder, image\_name)

if image\_path.endswith('.jpg') or image\_path.endswith('.png'):

row = df[df['ImageID'] == image\_name.split('.')[0]]

if not row.empty:

box = (row.iloc[0]['x\_center'], row.iloc[0]['y\_center'], row.iloc[0]['width'], row.iloc[0]['height'])

if os.path.exists(image\_path):

image, labeled\_mask, labeled\_mask\_fcm, labeled\_mask\_fcm\_improve, segmented\_image\_ssfcm, segmented\_image\_cs3fcm, segmented\_image\_ts3fcm = segment\_image\_with\_box(image\_path, box, n\_clusters)

display\_segmented\_image(image, labeled\_mask, labeled\_mask\_fcm, labeled\_mask\_fcm\_improve, segmented\_image\_ssfcm, segmented\_image\_cs3fcm, segmented\_image\_ts3fcm)

else:

print(f"Image {image\_name} not found in {image\_folder}")

else:

print('Row empty.')

### 5.3 Gui ( Giao diện)

* **def convert\_normalized\_box\_to\_pixel(box, image\_width, image\_height):**

x\_center, y\_center, width, height = box

x\_min = int((x\_center - width / 2) \* image\_width)

y\_min = int((y\_center - height / 2) \* image\_height)

x\_max = int((x\_center + width / 2) \* image\_width)

y\_max = int((y\_center + height / 2) \* image\_height)

return x\_min, y\_min, x\_max, y\_max

* **def create\_mask\_from\_box(image\_height, image\_width, box):**

mask = np.zeros((image\_height, image\_width), dtype=np.uint8)

x\_min, y\_min, x\_max, y\_max = box

mask[y\_min:y\_max, x\_min:x\_max] = 1

return mask

* **def create\_labeled\_mask\_fcm(image\_path, box):**

image = Image.open(image\_path)

image\_width, image\_height = image.size

pixel\_box = convert\_normalized\_box\_to\_pixel(box, image\_width, image\_height)

x\_min, y\_min, x\_max, y\_max = pixel\_box

cropped\_image = np.array(image)[y\_min:y\_max, x\_min:x\_max]

cropped\_image\_gray = cv2.cvtColor(cropped\_image, cv2.COLOR\_BGR2GRAY)

labeled\_data = []

labels = []

for y in range(y\_min, y\_max):

for x in range(x\_min, x\_max):

labeled\_data.append([y, x, cropped\_image\_gray[y - y\_min, x - x\_min]])

labels.append(1)

labeled\_data = np.array(labeled\_data)

labels = np.array(labels)

start\_time = time.time()

fcm = FuzzyCMeans(n\_clusters=2, m=2, max\_iter=100, tol=1e-4)

fcm.fit(cropped\_image\_gray.reshape(-1, 1))

fcm\_time = time.time() - start\_time

start\_time = time.time()

fcm\_improve = FuzzyCMeansImprove(n\_clusters=2, m=2, max\_iter=100, tol=1e-4)

fcm\_improve.fit(cropped\_image\_gray.reshape(-1, 1))

fcm\_improve\_time = time.time()-start\_time

labels = fcm.predict(cropped\_image\_gray.reshape(-1, 1)).reshape(cropped\_image\_gray.shape)

mask = (labels == 1).astype(np.uint8) # Giả sử nhãn '1' là tàu

labels\_improve = fcm\_improve.predict(cropped\_image\_gray.reshape(-1, 1)).reshape(cropped\_image\_gray.shape)

mask\_improve = (labels\_improve == 1).astype(np.uint8) # Giả sử nhãn '1' là tàu

full\_mask = np.zeros((image\_height, image\_width), dtype=np.uint8)

full\_mask[y\_min:y\_max, x\_min:x\_max] = mask

full\_mask\_improve = np.zeros((image\_height, image\_width), dtype=np.uint8)

full\_mask\_improve[y\_min:y\_max, x\_min:x\_max] = mask\_improve

times ={

'fcm\_time': fcm\_time,

'fcm\_improve\_time': fcm\_improve\_time

}

return full\_mask, full\_mask\_improve,times

* **def segment\_image\_with\_box(image\_path, box, n\_clusters=2):**

image = io.imread(image\_path)

image\_lab = color.rgb2lab(image)

image\_height, image\_width = image.shape[:2]

pixel\_box = convert\_normalized\_box\_to\_pixel(box, image\_width, image\_height)

labeled\_mask = create\_mask\_from\_box(image\_height, image\_width, pixel\_box)

labeled\_pixel\_index = np.where(labeled\_mask.flatten() == 1)[0]

image\_data = image\_lab.reshape((-1, 3)).astype(np.float32) / 255.0

labeled\_mask\_fcm, labeled\_mask\_fcm\_improve,times = create\_labeled\_mask\_fcm(image\_path, box)

start\_time = time.time()

sscfm = SSFCM(n\_clusters=n\_clusters, m=2, alpha=0.5, max\_iter=100, tol=0.0001, check=0)

sscfm.fit(image\_data, labeled\_pixel\_index)

sscfm\_time = time.time() - start\_time

start\_time = time.time()

cs3fcm = SSFCM(n\_clusters=n\_clusters, m=2, alpha=0.5, max\_iter=100, tol=0.0001, check=1)

cs3fcm.fit(image\_data, labeled\_mask\_fcm)

cs3fcm\_time = time.time() - start\_time + times["fcm\_time"]

start\_time = time.time()

ts3fcm = SSFCM(n\_clusters=n\_clusters, m=2, alpha=0.5, max\_iter=100, tol=0.0001, check=2)

ts3fcm.fit(image\_data, labeled\_mask\_fcm\_improve)

ts3fcm\_time =time.time() - start\_time + times["fcm\_improve\_time"]

times ={

'sscfm\_time': sscfm\_time,

'cs3fcm\_time': cs3fcm\_time,

'ts3fcm\_time': ts3fcm\_time

}

segmented\_image\_ssfcm = np.argmax(sscfm.U, axis=1).reshape(image\_lab.shape[:2])

segmented\_image\_cs3fcm = np.argmax(cs3fcm.U, axis=1).reshape(image\_lab.shape[:2])

segmented\_image\_ts3fcm = np.argmax(ts3fcm.U, axis=1).reshape(image\_lab.shape[:2])

return image, labeled\_mask\_fcm, labeled\_mask\_fcm\_improve, segmented\_image\_ssfcm, segmented\_image\_cs3fcm, segmented\_image\_ts3fcm,times

* **def display\_segmented\_image(image, labeled\_mask\_fcm, labeled\_mask\_fcm\_improve, segmented\_image\_ssfcm, segmented\_image\_cs3fcm, segmented\_image\_ts3fcm,times):**

contour\_image\_ssfcm = image.copy()

contour\_image\_cs3fcm = image.copy()

contour\_image\_ts3fcm = image.copy()

for label in np.unique(segmented\_image\_ssfcm):

mask = (segmented\_image\_ssfcm == label).astype(np.uint8)

contours, \_ = cv2.findContours(mask, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

cv2.drawContours(contour\_image\_ssfcm, contours, -1, (255, 0, 0), 1)

for label in np.unique(segmented\_image\_cs3fcm):

mask = (segmented\_image\_cs3fcm == label).astype(np.uint8)

contours, \_ = cv2.findContours(mask, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

cv2.drawContours(contour\_image\_cs3fcm, contours, -1, (255, 0, 0), 1)

for label in np.unique(segmented\_image\_ts3fcm):

mask = (segmented\_image\_ts3fcm == label).astype(np.uint8)

contours, \_ = cv2.findContours(mask, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

cv2.drawContours(contour\_image\_ts3fcm, contours, -1, (255, 0, 0), 1)

ax3 = fig.add\_subplot(3, 3, 4)

ax4 = fig.add\_subplot(3, 3, 5)

ax5 = fig.add\_subplot(3, 3, 7)

ax6 = fig.add\_subplot(3, 3, 8)

ax7 = fig.add\_subplot(3, 3, 9)

# Hiển thị hình ảnh và tiêu đề tương ứng cho từng trục

ax3.imshow(labeled\_mask\_fcm, cmap='gray')

ax3.set\_title('Labeled Mask FCM')

ax3.axis('off')

ax4.imshow(labeled\_mask\_fcm\_improve, cmap='gray')

ax4.set\_title('Labeled Mask FCM Improve')

ax4.axis('off')

ax5.imshow(contour\_image\_ssfcm)

ax5.set\_title(f'SSFCM (Time: {times["sscfm\_time"]:.2f}s)')

ax5.axis('off')

ax6.imshow(contour\_image\_cs3fcm)

ax6.set\_title(f'CS3FCM (Time: {times["cs3fcm\_time"]:.2f}s)')

ax6.axis('off')

ax7.imshow(contour\_image\_ts3fcm)

ax7.set\_title(f'TS3FCM (Time:{times["ts3fcm\_time"]:.2f}s)')

ax7.axis('off')

# Tạo bố cục chặt chẽ

fig.tight\_layout()

canvas.draw()

* **def dislay\_img\_label(image\_path,box):**

fig.clear()

image = io.imread(image\_path)

image\_height, image\_width = image.shape[:2]

pixel\_box = convert\_normalized\_box\_to\_pixel(box, image\_width, image\_height)

labeled\_mask = create\_mask\_from\_box(image\_height, image\_width, pixel\_box)

ax1 = fig.add\_subplot(3, 3, 1)

ax2 = fig.add\_subplot(3, 3, 2)

ax1.imshow(image)

ax1.set\_title('Original Image')

ax1.axis('off')

ax2.imshow(labeled\_mask, cmap='gray')

ax2.set\_title('Labeled Mask')

ax2.axis('off')

canvas.draw()

# Chuyển hướng stdout và stderr vào Text widget

* **class ConsoleOutput:**

def \_\_init\_\_(self, text\_widget):

self.text\_widget = text\_widget

* **def write(self, message):**

self.text\_widget.insert(tk.END, message)

self.text\_widget.see(tk.END)

* **def flush(self):**

pass # Không làm gì cả, chỉ để giữ tính tương thích

# Tkinter GUI

root = tk.Tk()

root.title("Image Segmentation")

# Tạo frame chính cho giao diện

main\_frame = tk.Frame(root)

main\_frame.pack(fill=tk.BOTH, expand=True)

# Tạo frame bên trái cho hình ảnh phân đoạn

left\_frame = tk.Frame(main\_frame, width=600, height=500)

left\_frame.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

fig = Figure(figsize=(15, 7))

canvas = FigureCanvasTkAgg(fig, master=left\_frame)

canvas.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

# Tạo frame bên phải cho các thành phần giao diện khác

right\_frame = tk.Frame(main\_frame, width=300, height=500)

right\_frame.pack(side=tk.RIGHT, fill=tk.Y, expand=False)

# Tạo frame con bên trong right\_frame cho các thành phần điều khiển phía trên

control\_frame = tk.Frame(right\_frame)

control\_frame.pack(side=tk.TOP, fill=tk.X, expand=False)

# Thành phần chọn ảnh

label\_path = tk.StringVar()

label = tk.Label(control\_frame, text="Image Path:")

label.pack(side=tk.TOP)

entry = tk.Entry(control\_frame, textvariable=label\_path)

entry.pack(side=tk.TOP)

csv\_path = 'Label\_img.csv'

df = pd.read\_csv(csv\_path)

file\_name\_label = tk.Label(control\_frame, text="File Name: N/A")

file\_name\_label.pack(side=tk.TOP)

* **def select\_image():**

file\_path = filedialog.askopenfilename(filetypes=[("Image files", "\*.jpg;\*.png")])

label\_path.set(file\_path)

file\_name = os.path.basename(file\_path)

file\_name\_label.config(text=f"File Name: {file\_name}")

row = df[df['ImageID'] == file\_name.split('.')[0]]

box = (row.iloc[0]['x\_center'], row.iloc[0]['y\_center'], row.iloc[0]['width'], row.iloc[0]['height'])

dislay\_img\_label(file\_path,box)

button = tk.Button(control\_frame, text="Select Image", command=select\_image)

button.pack(side=tk.TOP, padx=5, pady=5)

# Thành phần nhập số lượng cluster

cluster\_label = tk.Label(control\_frame, text="Number of Clusters:")

cluster\_label.pack(side=tk.TOP)

cluster\_entry = tk.Entry(control\_frame)

cluster\_entry.pack(side=tk.TOP)

* **def show\_error\_message(message):**

messagebox.showerror("Error", message)

# Nút thực hiện phân đoạn

* **def segment\_image():**

image\_path = label\_path.get()

if not image\_path.strip():

show\_error\_message("Please choose a picture")

return

if not cluster\_entry.get().strip(): # Kiểm tra xem cluster\_value có rỗng không sau khi loại bỏ khoảng trắng

show\_error\_message("Cluster entry is empty.")

return

n\_clusters = int(cluster\_entry.get())

if n\_clusters < 2 :

show\_error\_message("Number of clusters must be at least 2.")

return

file\_name = os.path.basename(image\_path)

file\_name\_label.config(text=f"File Name: {file\_name}")

box = None

row = df[df['ImageID'] == file\_name.split('.')[0]]

box = (row.iloc[0]['x\_center'], row.iloc[0]['y\_center'], row.iloc[0]['width'], row.iloc[0]['height'])

image,labeled\_mask\_fcm, labeled\_mask\_fcm\_improve, segmented\_image\_ssfcm, segmented\_image\_cs3fcm, segmented\_image\_ts3fcm,times = segment\_image\_with\_box(image\_path, box, n\_clusters)

display\_segmented\_image(image, labeled\_mask\_fcm, labeled\_mask\_fcm\_improve, segmented\_image\_ssfcm, segmented\_image\_cs3fcm, segmented\_image\_ts3fcm,times)

button\_segment = tk.Button(control\_frame, text="Segment", command=segment\_image)

button\_segment.pack(side=tk.TOP, padx=5, pady=5)

# Tạo Text widget cho console bên dưới

console\_frame = tk.Frame(right\_frame)

console\_frame.pack(side=tk.BOTTOM, fill=tk.BOTH, expand=True)

console = tk.Text(console\_frame, wrap='word', height=15)

console.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

scrollbar = tk.Scrollbar(console\_frame, command=console.yview)

scrollbar.pack(side=tk.RIGHT, fill=tk.Y)

console['yscrollcommand'] = scrollbar.set

# Chuyển hướng stdout và stderr

sys.stdout = ConsoleOutput(console)

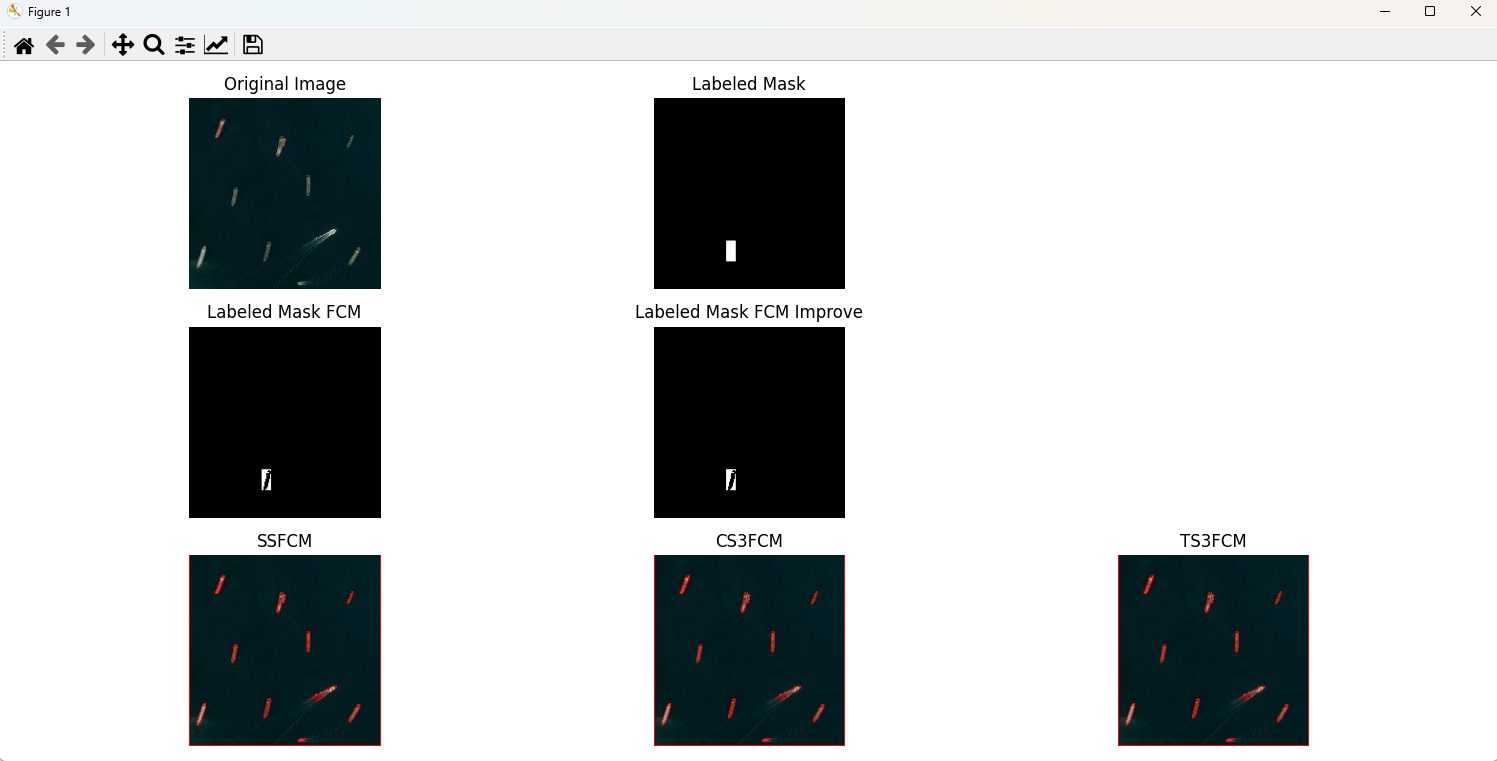
sys.stderr = ConsoleOutput(console)

root.mainloop()

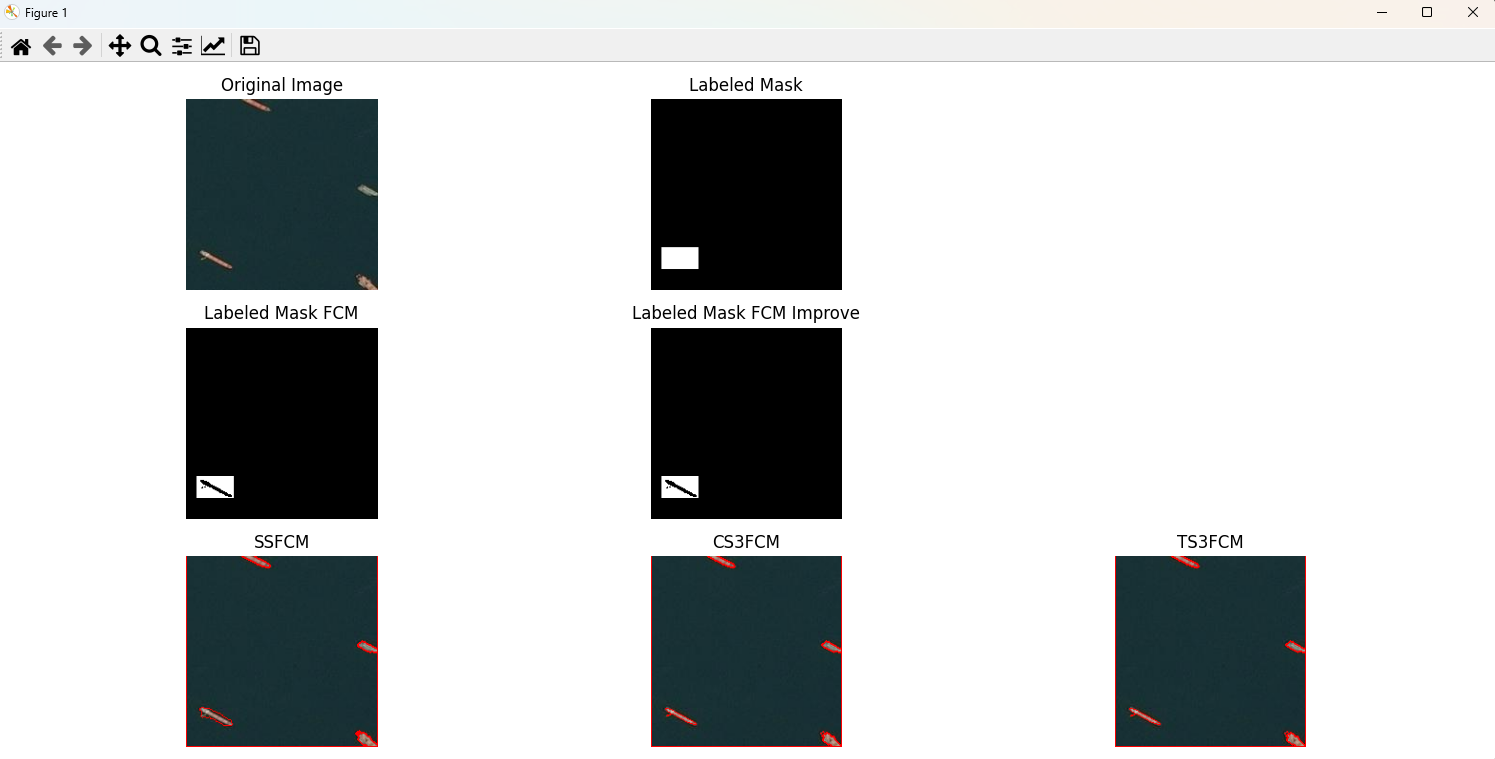
# IV. KẾT QUẢ

### 4.1 Main

**\* Ảnh 1: 0b1b3d75b.jpg**



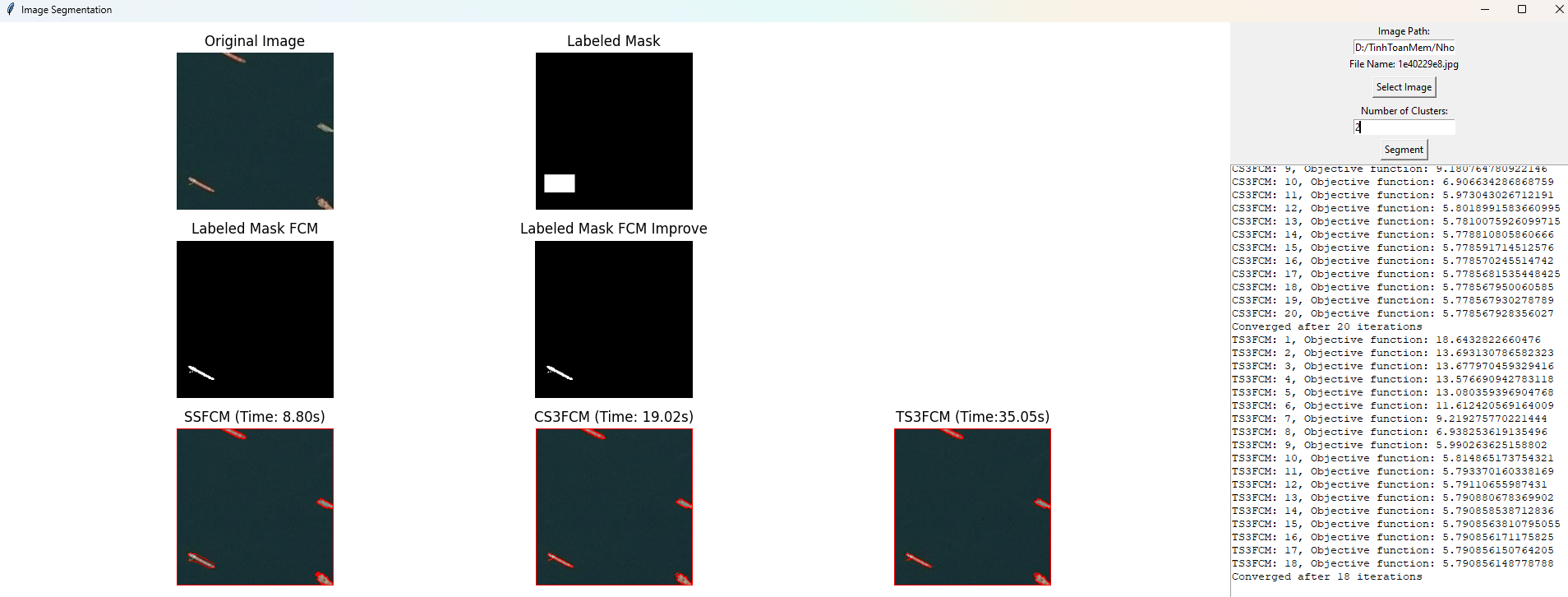
**\* Ảnh 2: 1e40229e8.jpg**



### 4.2 Gui ( giao diện )

**\* Số cụm đúng với ảnh:**

* Ảnh 1:

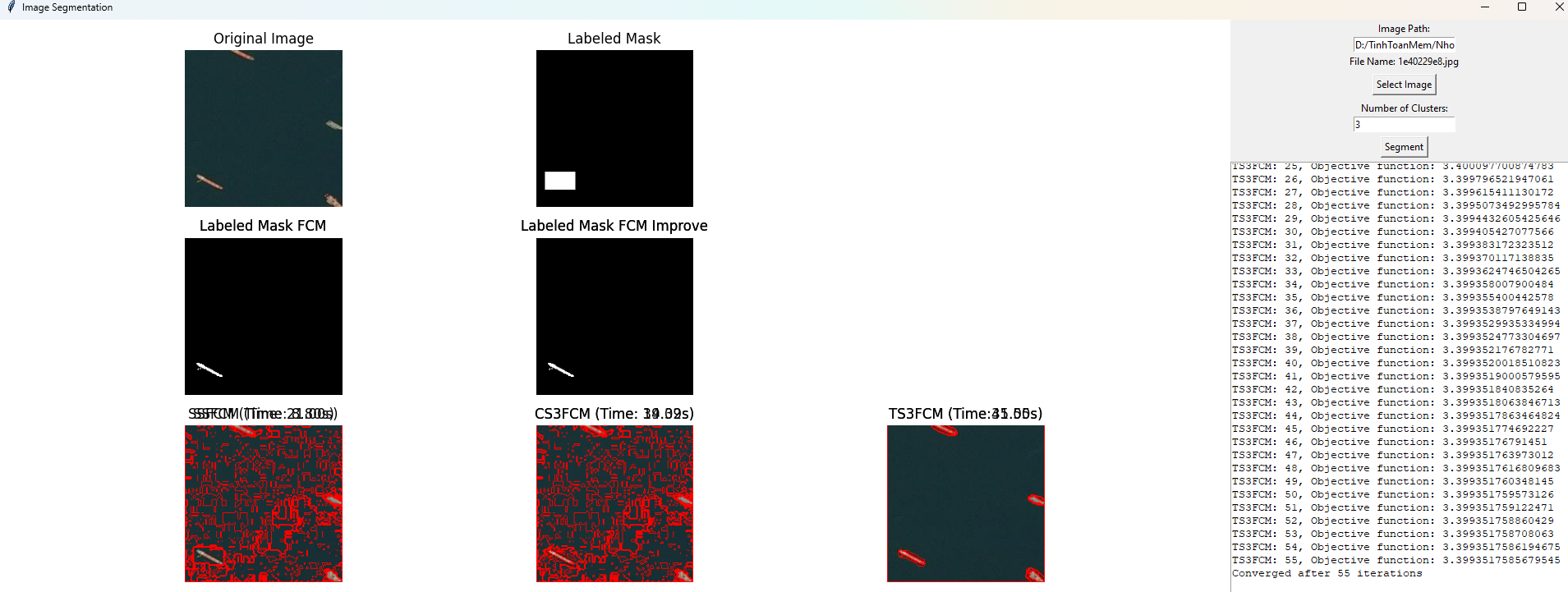


* Ảnh 2:

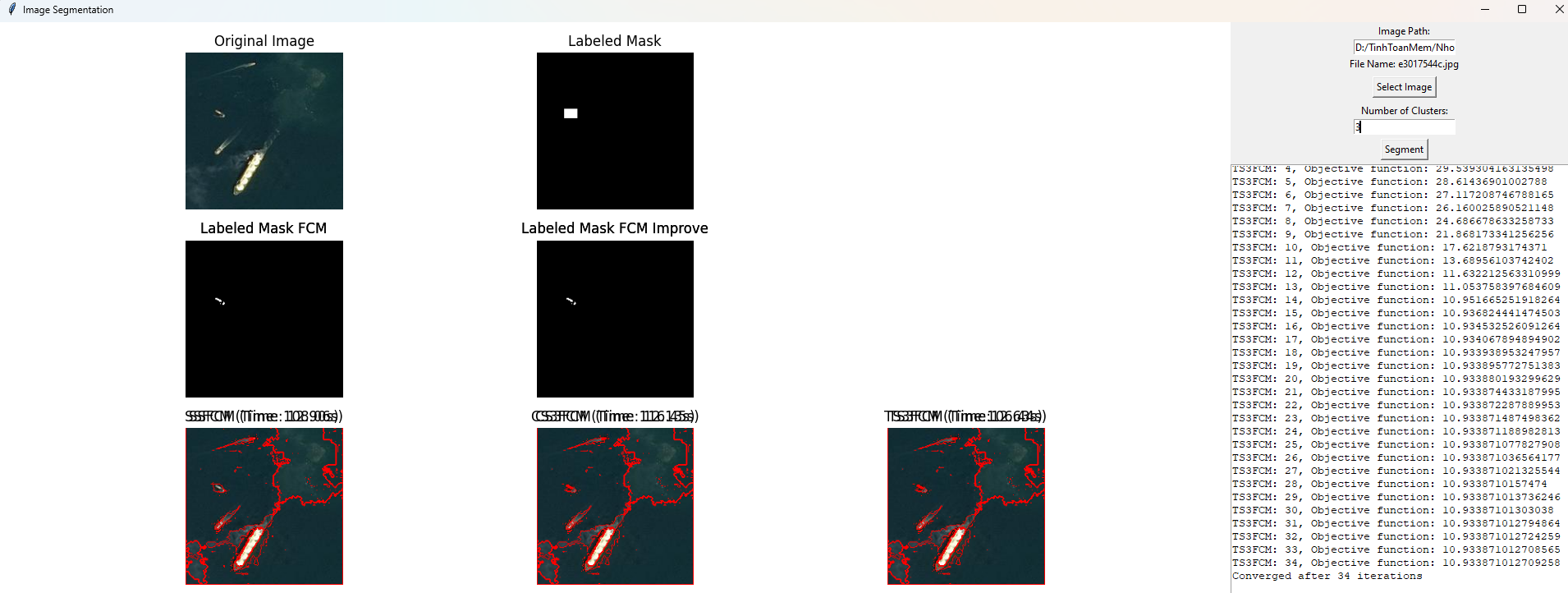


**\* Số cụm sai với ảnh:**

* Ảnh 1:



* Ảnh 2:



# V. KẾT LUẬN

Trong dự án "Phát hiện tàu biển từ ảnh vệ tinh dựa trên phương pháp bán giám sát mờ an toàn", chúng em đã sử dụng dữ liệu từ cuộc thi "Airbus Ship Detection" trên Kaggle, cung cấp một bộ dữ liệu phong phú và đa dạng về ảnh vệ tinh có độ phân giải cao. Qua quá trình phân tích dữ liệu, chúng em đã nhận diện được các đặc điểm quan trọng của dữ liệu, bao gồm phân phối kích thước tàu biển, môi trường xung quanh, và tính không đồng nhất của các ảnh vệ tinh. Điều này cho phép chúng em thiết lập một quy trình tiền xử lý dữ liệu hợp lý và hiệu quả, tạo điều kiện thuận lợi cho việc huấn luyện mô hình phát hiện tàu biển.

Việc sử dụng phương pháp bán giám sát mờ an toàn giúp tận dụng tối đa lượng dữ liệu không gán nhãn, cải thiện độ chính xác của mô hình mà không đòi hỏi một lượng lớn dữ liệu có gán nhãn chi tiết. Phương pháp này không chỉ giúp tiết kiệm thời gian và chi phí trong việc thu thập và gán nhãn dữ liệu, mà còn tăng cường khả năng tổng quát hóa của mô hình đối với các tình huống mới và phức tạp.

Dự án này không chỉ đóng góp vào lĩnh vực phát hiện tàu biển từ ảnh vệ tinh, mà còn mở ra các hướng nghiên cứu mới trong việc áp dụng các phương pháp học máy bán giám sát cho các bài toán nhận dạng và phát hiện đối tượng trong ảnh vệ tinh. Kết quả đạt được từ dự án có thể được ứng dụng trong nhiều lĩnh vực khác nhau, từ giám sát hàng hải, an ninh quốc phòng, đến bảo vệ môi trường biển.

Chúng em nhận thấy rằng, mặc dù dự án đã đạt được một số kết quả tích cực, nhưng vẫn còn nhiều hạn chế và thách thức cần khắc phục. Chúng em hy vọng rằng những nỗ lực này sẽ góp phần vào việc phát triển các giải pháp giám sát biển tiên tiến và hiệu quả hơn trong tương lai. Chúng em rất mong nhận được những góp ý và hướng dẫn từ các thầy cô để có thể hoàn thiện hơn nữa dự án của mình.

# 

# TÀI LIỆU THAM KHẢO

* Bài giảng Khai phá dữ liệu - thầy **Trần Mạnh Tuấn**
* <http://www.thuvienlamdong.org.vn:81/bitstream/DL_134679/52688/1/CTv178V227S082022103.pdf>
* <https://tailieu.vn/doc/luan-an-tien-si-khoa-hoc-may-tinh-nghien-cuu-phat-trien-phuong-phap-phan-cum-ban-giam-sat-mo-an-toa-2701220.html>
* <https://www.kaggle.com/competitions/airbus-ship-detection/overview>
* <https://www.kaggle.com/c/airbus-ship-detection/>
* <https://github.com/HumanSignal/labelImg>
* <https://www.youtube.com/watch?v=de3issSnfKI&t=7s>