|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **BỘ KHOA HỌC**  **VÀ CÔNG NGHỆ** | **VIỆN HÀN LÂM KHOA HỌC**  **VÀ CÔNG NGHỆ VIỆT NAM** | | **CHƯƠNG TRÌNH KHOA HỌC VÀ CÔNG NGHỆ CẤP QUỐC GIA**  **VỀ CÔNG NGHỆ VŨ TRỤ (2016 – 2020)** | | |  | |   **BÁO CÁO CÔNG VIỆC SỐ 6.6.2**  **“XÂY DỰNG MÔ ĐUN SỬ DỤNG PHƯƠNG PHÁP CẢI TIẾN RXD”**  **ĐỀ TÀI: “Nghiên cứu, xây dựng hệ thống tích hợp thông tin đa nguồn (ảnh viễn thám, AIS, LRIT ...) phục vụ công tác bảo đảm an ninh quốc gia”**  **Mã số: VT-UD.06/16-20**   |  |  |  |  | | --- | --- | --- | --- | | 1. **Cơ quan chủ trì đề tài:** | | | 1. **Cục B42, Tổng cục V, Bộ Công an** | | 1. **Chủ nhiệm đề tài:** | | 1. **Lương Nguyễn Hoàng Hoa** | | | 1. **Người thực hiện:** 2. **Trần Kim Hoàn** 3. **Mẫn Đức Chức** 4. **Phạm Văn Hà** 5. **Nguyễn Văn Hải** 6. **Trần Tuấn Vinh** 7. **Cao Hồng Huệ** | **Nguyễn Thị Nhật Thanh**  **Lưu Việt Hưng**  **Lưu Quang Thắng**  **Hoàng Xuân Phương**  **Phan Anh**  **Hà Đức Văn**  **Bùi Quang Hưng**  **Nguyễn Quang Thành**  **Nguyễn Huy Thảo** | | |     **Hà Nội –2018** | |
| |  |  | | --- | --- | | **BỘ KHOA HỌC**  **VÀ CÔNG NGHỆ** | **VIỆN HÀN LÂM KHOA HỌC**  **VÀ CÔNG NGHỆ VIỆT NAM** | | **CHƯƠNG TRÌNH KHOA HỌC VÀ CÔNG NGHỆ CẤP QUỐC GIA**  **VỀ CÔNG NGHỆ VŨ TRỤ (2016 – 2020)** | | |  | |     **BÁO CÁO CÔNG VIỆC SỐ 6.6.2**  **“XÂY DỰNG MÔ ĐUN SỬ DỤNG PHƯƠNG PHÁP CẢI TIẾN RXD”**  **ĐỀ TÀI: “Nghiên cứu, xây dựng hệ thống tích hợp thông tin đa nguồn (ảnh viễn thám, AIS, LRIT ...) phục vụ công tác bảo đảm an ninh quốc gia”**  **Mã số: VT-UD.06/16-20**  **Người thực hiện: Bùi Quang Hưng**  **Trần Kim Hoàn**  **Hà Nội – 2018** | |

# MỤC LỤC

[MỤC LỤC 1](#_Toc519181828)

[DANH MỤC CÁC CHỮ VIẾT TẮT 3](#_Toc519181829)

[DANH MỤC CÁC BẢNG 4](#_Toc519181830)

[DANH MỤC CÁC HÌNH 5](#_Toc519181831)

[MỞ ĐẦU 6](#_Toc519181832)

[CHƯƠNG 1. GIỚI THIỆU VỀ PHƯƠNG PHÁP 7](#_Toc519181833)

[1.1. Tổng quan phương pháp 7](#_Toc519181834)

[1.2. Pansharpen dữ liệu NIR và PAN 7](#_Toc519181835)

[1.3. Phân tích thành phần chính 8](#_Toc519181836)

[1.4. Bản đồ giá trị dị thường 10](#_Toc519181837)

[1.4.1. Giá trị cường độ bất thường toàn cục 10](#_Toc519181838)

[1.4.2. Giá trị bất thường texture 11](#_Toc519181839)

[1.4.3. Bản đồ dị thường tổng hợp 13](#_Toc519181840)

[CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ XÂY DỰNG MÔ ĐUN SỬ DỤNG PHƯƠNG PHÁP CẢI TIẾN RXD 14](#_Toc519181841)

[2.1. Đặc tả yêu cầu người dùng 15](#_Toc519181845)

[2.1.1. Đặc tả yêu cầu chức năng 15](#_Toc519181846)

[2.1.2. Đặc tả yêu cầu phi chức năng 16](#_Toc519181847)

[2.2. Đặc tả ca sử dụng 16](#_Toc519181848)

[2.2.1. Xây dựng biểu đồ ca sử dụng 16](#_Toc519181849)

[2.2.2. Phân tích luồng cơ bản 17](#_Toc519181850)

[2.2.3. Luồng rẽ nhánh 18](#_Toc519181851)

[2.3. Xây dựng biểu đồ lớp 20](#_Toc519181852)

[2.4. Xây dựng biểu đồ tuần tự 21](#_Toc519181853)

[2.5. Phân tích thành phần Mã nguồn xử lý thuật toán 23](#_Toc519181854)

[CHƯƠNG 3. XÂY DỰNG MÔ-ĐUN SỬ DỤNG PHƯƠNG PHÁP CẢI TIẾN RXD 24](#_Toc519181855)

[3.1. Trình diễn mã nguồn 24](#_Toc519181856)

[3.1.1. Mã nguồn xử lý thuật toán 24](#_Toc519181857)

[3.1.2. Mã nguồn Thành phần thực thi 27](#_Toc519181858)

[3.1.3. Mã nguồn giao diện tương tác 28](#_Toc519181859)

[3.2. Trình diễn mô-đun 30](#_Toc519181860)

[KẾT LUẬN 35](#_Toc519181861)

[TÀI LIỆU THAM KHẢO 36](#_Toc519181862)

# DANH MỤC CÁC CHỮ VIẾT TẮT

|  |  |
| --- | --- |
| **Từ/Cụm từ viết tắt** | **Giải thích** |
| CSDL | Cơ sở dữ liệu |
| EIN | Effective Intensity Number |
| HDFS | Hadoop File System |
| MIN | Majority Intensity Number |
| RDD | Resilient Distributed Dataset |
| RXD | Reed-Xiaoli Detector |
| VHR | Very High Resolution |

# DANH MỤC CÁC BẢNG

[Bảng 2.1: Các lớp trong ca sử dụng 23](#_Toc519167318)

# DANH MỤC CÁC HÌNH

[Hình 1.1: Tổng quan phương pháp 7](#_Toc519171416)

[Hình 1.2. Tính giá trị Gradient 11](#_Toc519171417)

[Hình 1.3 Ước lượng lỗi khi tính giá trị góc của gradient do nhiễu lượng hóa (von Gioi et al., 2010) 13](#_Toc519171418)

[Hình 2.1: Kiến trúc Bộ công cụ phân tích và xử lý dữ liệu ảnh quang học và radar 14](#_Toc519171419)

[Hình 2.2. Biểu đồ ca sử dụng 17](#_Toc519171420)

[Hình 2.3 Biểu đồ lớp của ca sử dụng 20](#_Toc519171421)

[Hình 2.4 Biểu đồ tuần tự của ca sử dụng 22](#_Toc519171422)

[Hình 3.1: quản lý dữ liệu và tải ảnh vệ tinh lên bộ công cụ 31](#_Toc519171423)

[Hình 3.2: Mô-đun sử dụng phương pháp RXD cải tiến 31](#_Toc519171424)

[Hình 3.3: Lựa chọn ảnh đầu vào 32](#_Toc519171425)

[Hình 3.4: Mô-đun đã nhận dữ liệu ảnh đầu vào 32](#_Toc519171426)

[Hình 3.5: Quá trình xử lý đang diễn ra 33](#_Toc519171427)

[Hình 3.6: Nhật ký thực thi chương trình 33](#_Toc519171428)

[Hình 3.7: Kết quả xử lý 34](#_Toc519171429)

[Hình 3.8: Quản lý lịch sử của mô-đun 34](#_Toc519171430)

# MỞ ĐẦU

Hiện nay, phát hiện hoạt động của tàu thuyền đang là một vấn đề nhận được nhiều sự quan tâm từ cá nhà quản lý ở Việt Nam. Việc theo dõi và đánh giá hoạt động của tàu thuyền ở những khu vực cảng biển, hải đảo đã góp phần không nhỏ trong phát triển kinh tế, đồng thời đảm bảo an ninh quốc phòng.

Bài toán phát hiện tàu gồm 3 bước:

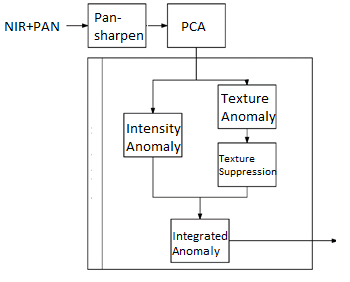
* Phát hiện điểm dị thường.
* Trích chọn đối tượng dị thường.
* Phân tích (phân lớp) để phát hiện tàu.

Với các nguồn ảnh vệ tinh sẽ có những ứng dụng khác nhau và được phân tích và xử lý bằng những thuật toán khác nhau. Trong bước phát hiện điểm dị thường có nhiều phương pháp khác nhau để có thể thực hiện phát hiện điểm dị thường. Trong đó phương pháp RXD (Reed-Xiaoli Detector) là một trong những phương pháp phát hiện các đối tượng dị thường trên biển được sử dụng rộng rãi. RXD là Thuật toán phát hiện bất thường được xử dụng rộng rãi. RXD là một phương pháp thống kê để phát hiện pixel trong hình ảnh hyperspectral khác biệt đáng kể so với các pixel khác trong miền địa phương của chúng. Thuật toán RXD dựa trên giả định của một mẫu thử rời rạc thống nhất hiện có trong cả không gian và phổ. Tuy nhiên, phương pháp RXD cơ bản còn nhiều yếu độ như chưa thể hện rõ sự khác biệt giữa các điểm ảnh và độ phức tạp tính toán lớn. Trong chuyên đề này, chúng tôi xin được trình bày về việc xây dựng mô đun sử dụng phương pháp cải tiến RXD.

# GIỚI THIỆU VỀ PHƯƠNG PHÁP

## Tổng quan phương pháp

Tổng quan các bước phương pháp RXD cải tiến được trình bày như trong Hình 1. Cụ thể, phương pháp đề xuất bao gồm hai bước chính. Đầu tiên, dữ liệu band Near Infrared được tăng cường độ phân giải sử dụng phương pháp pan-sharpen. Ảnh tăng cường độ phân giải band Near Infrared (gọi tắt là PNIR) sẽ được tích hợp với dữ liệu ảnh Panchromatic (PAN) gốc sử dụng phương pháp Principle Component Analysis (PCA). Thành phần đầu tiền của PCA sau đó sẽ được sử dụng làm đầu vào cho bước tính giá trị dị thường. Tại bước tính giá trị dị thường, hai loại giá trị dị thường được sử dụng để đánh giá các điểm ảnh bao gồm intensity anomaly và texture anomaly. Sự kết hợp của hai loại giá trị dị thường này sẽ cho chúng ta bản đồ dị thường với cái nhìn đầy đủ từ tổng quan (global) đến cục bộ (local) sự khác biệt của điểm ảnh đang xét.



Hình 1.1: Tổng quan phương pháp

## Pansharpen dữ liệu NIR và PAN

Do các vấn đề chi phí và độ phức tạp, các vệ tinh độ phân giải siêu cao (Very High Resolution -VHR) thường cung cấp cho chúng ta hình ảnh PAN với độ phân giải không gian tốt hơn so với hình ảnh đa phổ. Tuy nhiên, hình ảnh đa phổ lại có độ phân giải phổ cao hơn so với hình ảnh PAN, do đó được sử dụng nhiều cho các bài toán phân loại đối tượng dựa trên phổ. Sự kết hợp tốt của ảnh đa phổ và PAN có thể tận dụng lợi thế của cả hai. Một dữ liệu kết hợp có thể giữ độ phân giải phổ của ảnh đa phổ và độ phân giải không gian của ảnh PAN (Luu, Pham, Man, Bui, & Nguyen, 2016).

Phương pháp UNB pan-sharp được đề xuất trong (Y. Zhang, n.d.)[7] là một thuật toán hoàn toàn tự động đã được sử dụng trong nhiều phần mềm thương mại, ví dụ: PCI Geomatics, Digital Globe. Thuật toán sử dụng phương pháp bình phương tối thiểu để tìm ra sự phù hợp tốt nhất giữa các giá trị xám độ của ảnh PAN và các dữ liệu đa phổ để điều chỉnh sự đóng góp của các band dữ liệu đa phổ riêng lẻ (Yun Zhang & Mishra, 2013). Ảnh hưởng của sự biến thiên dữ liệu trong ảnh tổng hợp có thể được loại bỏ bằng một tập hợp các phương pháp thống kê để thiết lập mối quan hệ giữa các giá trị xám độ của các band khác nhau. Do đó, không có bất kỳ tham số nào do người dùng chỉ định, UNB sẽ tạo ra các kết quả hợp nhất chất lượng cao phù hợp bất kể biến thể của cảm biến và hình ảnh.

## Phân tích thành phần chính

Sử dụng phương pháp UNB pan-sharp chúng ta sẽ có dữ liệu ảnh NIR với độ phân giải tương đương dữ liệu PAN. Trong phần này, dữ liệu NIR tăng cường sẽ được kết hợp với PAN thành một ảnh duy nhất sử dụng phương pháp phân tích thành phần chính.

Phương pháp phân tích thành phần chính đầu tiên sẽ biến đổi mỗi ảnh đa phổ , với , , là chiều dài, rộng và số lượng band tương ứng, thành vector có dạng:

|  |  |
| --- | --- |
|  | (1) |

Với là giá trị của các band tại một vị trí *i*. Vector trung bình được tính như sau:

|  |  |
| --- | --- |
|  | (2) |

Ma trận hiệp phương sai của được định nghĩa như sau:

|  |  |
| --- | --- |
|  | (3) |

Các vector riêng tương ứng với các giá trị riêng của ma trận hiệp phương sai thỏa mãn:

|  |  |
| --- | --- |
|  | (4) |

Trong đó là ma trận chéo với giá trị là các giá trị riêng , và là ma trận trực giao chứa các vector riêng *K* chiều.

Trong (Rodarmel & Shan, 2002), các tác giả cho rằng lượng thông tin chứa trong các thành phần chính giảm dần khi số lượng thành phần tăng lên và hầu hết thông tin của dữ liệu sẽ được thể hiện trong thành phần chính đầu tiên. Do đó, thành phần chính của phép phân tích thành phần chính giữa ảnh PAN và ảnh NIR tăng cường được sử dụng. Giả sử các giá trị riêng và vector riêng được xắp xếp giảm dần như sau , dòng đầu tiền của ma trận , hay còn gọi là vector riêng đầu tiên có thể được sử dụng để tính thành phần chính đầu tiên như sau:

|  |  |
| --- | --- |
|  | (5) |

trong đó là giá trị của điểm ảnh của thành phần chính đầu tiên, là vector giá trị nguyên gốc của các band ảnh tại điểm .

## Bản đồ giá trị dị thường

Trong ảnh quang học, tàu có thể được xem là các điểm ảnh bất thường trong nền đại dương đồng nhất (Guang Yang, Bo Li, Shufan Ji, Feng Gao, & Qizhi Xu, 2014) [6]. Do đó chúng có thể được phát hiện bằng cách tìm các điểm ảnh có độ sáng bất thường bằng cách so sánh cường độ gặp phải với các đặc tính thống kê của nền biển địa phương và toàn cục (Crisp, 2004)[2]. Trong nghiên cứu này, chúng tôi thực hiện theo phương pháp được đề xuất bởi (Guang Yang et al., 2014)[6] sử dụng giá trị cường độ bất thường toàn cục và giá trị kết cấu bất thường địa phương.

### Giá trị cường độ bất thường toàn cục

Do kích thước tàu thường nhỏ và nền biển chiếm phần chính của ảnh, tần số xuất hiện các giá trị xám độ của các điểm ảnh thuộc về mục tiêu tàu thường rất thấp. Do đó, bất thường toàn cục được định nghĩa để nhấn mạnh sự bất thường về tần xuất xuất hiện xám độ của con tàu như sau:

|  |  |
| --- | --- |
|  | (6) |

trong đó là tần suất xuất hiện giá trị xám độ của tàu trong ảnh.

### Giá trị bất thường texture

Không giống như bất thường toàn cục, bất thường địa phương được tính trong một khu vực địa phương xung quanh mục tiêu. Tàu có thể được coi là đối tượng dễ thấy trên nền biển, do đó có thể dễ dàng phát hiện với một phương pháp phát hiện cạnh.

|  |  |  |  |
| --- | --- | --- | --- |
| (**a**) | (**b)** | (**c**) | (**d**) |

Hình 1.2. Tính giá trị Gradient

Với hình ảnh , giá trị điểm ảnh tại vị trí được kí hiệu là (Hình 2b). Giá trị gradient tại điểm ảnh theo chiều ngang và dọc được tính bởi:

|  |  |
| --- | --- |
|  | (7) |
|  | (8) |

Giá trị bất thường địa phương được định nghĩa bởi:

|  |  |
| --- | --- |
|  | (9) |

Có thể nhận thấy, các điểm ảnh có giá trị gradient nhỏ thường tương ứng với các điểm nhiễu bề mặt biển. Ngoài ra, việc tính toán giá trị gradient cho các điểm ảnh này cũng thường có lỗi cao hơn do vấn đề lượng hóa hình ảnh (Agneès Desolneux Desolneux, Moisan, & Morel, 2001; Grompone von Gioi, Jakubowicz, Morel, & Randall, 2012; von Gioi, Jakubowicz, Morel, & Randall, 2010). Do đó, các điểm ảnh có giá trị gradient nhỏ hơn một ngưỡng sẽ được loại bỏ. Như có thể thấy trong Hình 2c and Hình 2d, các điểm ảnh với giá trị gradient nhỏ tạo bởi nhiễu được loại bỏ, trong khi đó các điểm ảnh tại cạnh thuyền được giữ lại.

Giá trị ngưỡng được tính dựa trên lý thuyết trình bày bởi (Agneès Desolneux, Ladjal, Moisan, & Morel, 2002), trong đó các tác giả cho thấy việc lượng hóa giá trị xám độ ảnh gây ra lỗi khi tính hướng của các điểm ảnh. Giá trị được thiết lập với giả thiết với các điểm ảnh có giá trị gradient lớn hơn , lỗi trong việc tính giá trị góc của gradient sẽ nhỏ hơn giá trị .

Gọi , và là ảnh, ảnh được lượng hóa và nhiễu khi lượng hóa, ta có:

|  |  |
| --- | --- |
|  | (10) |
|  | (11) |

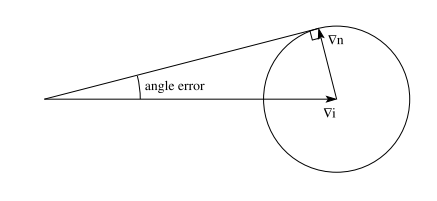
Trong nghiên cứu (Agneès Desolneux et al., 2002), lỗi của việc tính giá trị góc của gradient được cho bởi (Hình 3):

|  |  |
| --- | --- |
|  | (12) |

trong đó được thiết lập bởi . Với điều kiện , chúng ta có:

|  |  |
| --- | --- |
|  | (13) |

Giá trị ngưỡng được thiết lập dựa trên công thức (13) trong đó thể hiện khả năng lỗi cao nhất trong việc tính giá trị gradient và là dung sai góc. Dựa trên thực nghiệm, chúng tôi chọn và độ. Các tham số này cũng đồng nhất với giá trị được sử dụng trong (Burns, Hanson, & Riseman, 1986; von Gioi et al., 2010).



Hình 1.3 Ước lượng lỗi khi tính giá trị góc của gradient do nhiễu lượng hóa (von Gioi et al., 2010)

### Bản đồ dị thường tổng hợp

Bản đồ dị thường tổng hợp được tính như sau:

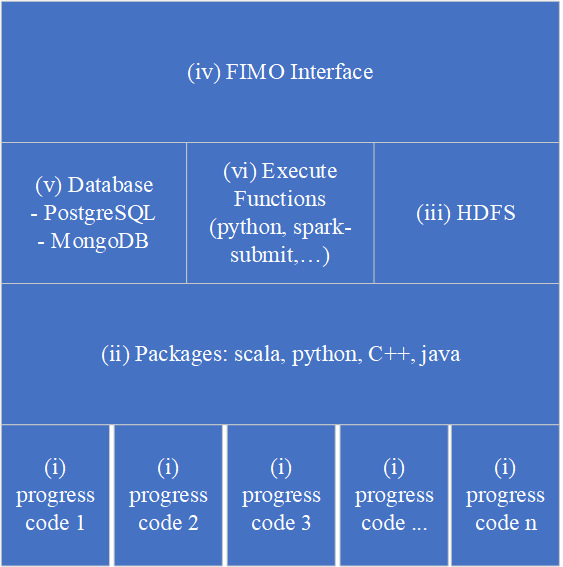
|  |  |
| --- | --- |
|  | (14) |

Trong đó và được chuẩn hóa về khoảng . Với việc sử dụng bản đồ dị thường tổng hợp, chúng tôi đảm bảo rằng các đối tượng được phát hiện thỏa mãn đồng thời hai điều kiện cùng lúc: các vùng điểm ảnh lớn (ví dụ như mây, đảo nổi, vùng nước bất thường, …) và nhiễu tạo ra bởi các điểm sáng bất thường tạo ra bởi sóng được loại bỏ.

# PHÂN TÍCH THIẾT KẾ XÂY DỰNG MÔ ĐUN SỬ DỤNG PHƯƠNG PHÁP CẢI TIẾN RXD

Mô-đun sử dụng phương pháp cải tiến RXD là một thành phần của Bộ công cụ phân tích và xử lý dữ liệu ảnh quang học và radar. Nghiên cứu Phân tích thiết kế xây dựng bộ công cụ phân tích và xử lý dữ liệu ảnh quang học và radar đã thu thập, đặc tả và phân tích các yêu cầu chức năng và phi chức năng của bộ công cụ. Từ đó, nghiên cứu Phân tích thiết kế xây dựng bộ công cụ phân tích và xử lý dữ liệu ảnh quang học và radar đã đề xuất thiết kế kiến trúc, thiết kế cơ sở dữ liệu và thiết kế khung giao diện chung cho bộ công cụ. Theo đó, bộ công cụ bao gồm 6 phần:

1. Thành phần mã nguồn của các thuật toán xử lý (progress code)
2. Thành phần đóng gói của các thuật toán xử lý (package)
3. Thành phần lưu trữ dữ liệu phân tán (HDFS)
4. Giao diện tương tác người dùng (interface)
5. Cơ sở dữ liệu (database)
6. Thành phần thực thi (execute function)



Hình 2.1: Kiến trúc Bộ công cụ phân tích và xử lý dữ liệu ảnh quang học và radar

Bộ công cụ được thiết kế theo mô hình mô đun để đảm bảo khả năng mở rộng, phát triển độc lập và dễ bảo trì. Thành phần (iii) và (v) được sử dụng để lưu trữ dữ liệu chung cho bộ công cụ. Thành phần (i), (ii), (iv), (vi) sẽ được xây dựng độc lập cho từng mô đun. Trong chương này, nhóm nghiên cứu sẽ phân tích, thiết kế để xây dựng Mô-đun sử dụng phương pháp cải tiến RXD. Phần 2.1 – 2.2 đặc tả các yêu cầu chuyên biệt của người sử dụng với mô đun. Phần 2.3 – 2.4 phân tích và thiết kế cách thức hoạt động của mô đun, cấu trúc dữ liệu cần sử dụng dựa trên cái thông tin đặc tả yêu cầu. Phần 2.5 mô tả danh sách tham số và công nghệ xây dựng, kết hợp của Mã nguồn xử lý và Thành phần thực thi.



## Đặc tả yêu cầu người dùng

Trong công nghệ phần mềm, thu thập yêu cầu là giai đoạn đầu tiên trong vòng đời phát triển của một sản phẩm (có thể là một phần mềm độc lập, một hệ thống hoặc một mô-đun của hệ thống). Yêu cầu người (user requirements) dùng được định nghĩa là các mô tả về chức năng và phi chức năng của sản phẩm ở mức độ khái quát, không bao gồm các thông tin kỹ thuật chi tiết. Bộ công cụ phân tích và xử lý dữ liệu ảnh quang học và radar cung cấp các công cụ tự động hóa xử lý ảnh viễn thám trong từng bước tiền xử lý và phát hiện, giám sát tàu thuyền và giàn khoan trên biển, hải đảo sử dụng ảnh vệ tinh radar. Mô-đun sử dụng phương pháp cải tiến RXD là một thành phần của bộ công cụ với chức năng cài đặt và cung cấp cách thức tương tác để sử dụng phương pháp với dữ liệu và tham số đầu vào tùy chỉnh. Tác nhân sử dụng của mô đun này được xác định là các chuyên gia xử lý dữ liệu.

### Đặc tả yêu cầu chức năng

Dựa trên danh sách công cụ, thu thập yêu cầu của chuyên gia có kinh nghiệm trong xử lý ảnh viễn thám và tham khảo các công cụ xử lý ảnh hiện ảnh, yêu cầu chức năng của bộ công cụ được mô tả như sau:

* + Chọn ảnh từ thành phần lưu trữ
  + Chọn thuật toán xử lý trong mô đun
  + Nhập các tham số mong muốn
  + Xem tiến độ xử lý
  + Phát hiện và báo lỗi trong quá trình xử lý
  + Xem nhật ký xử lý
  + Điều hướng tới kết quả sau xử lý

### Đặc tả yêu cầu phi chức năng

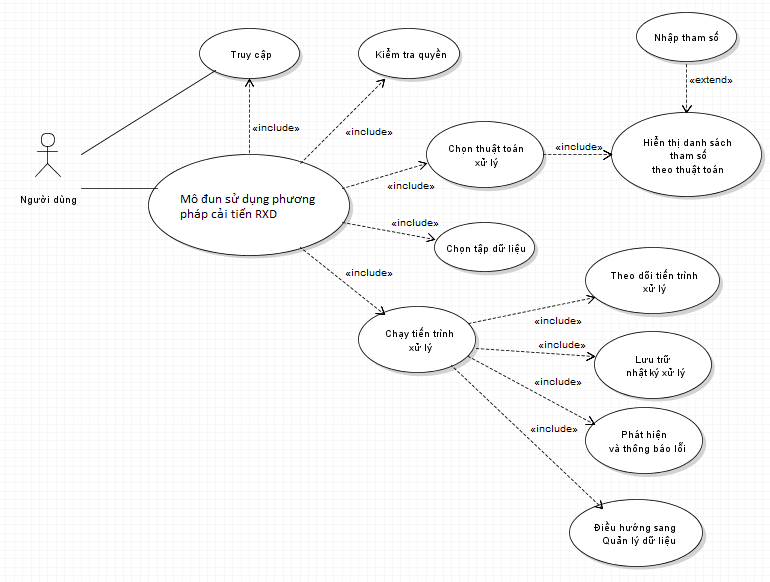
Dựa trên danh sách công cụ, thu thập yêu cầu của chuyên gia có kinh nghiệm trong xử lý ảnh viễn thám và tham khảo các công cụ xử lý ảnh hiện ảnh, yêu cầu phi chức năng của bộ công cụ được mô tả như sau:

* + Giao diện thân thiện
  + Hệ thống dễ sử dụng

## Đặc tả ca sử dụng

### Xây dựng biểu đồ ca sử dụng

Mô tả các ca sử dụng là một kỹ thuật khai phá yêu cầu được giới thiệu trong phương pháp Objectory bởi Jacobson, 1993. Ca sử dụng mô tả các tác nhân tham gia vào một tương tác trên phần mềm. Dựa trên các yêu cầu chức năng của bộ công cụ, quy trình tương tác chính của mô đun sử dụng phương pháp cải tiến RXD gồm các bước sau:



Hình 2.2. Biểu đồ ca sử dụng

### Phân tích luồng cơ bản

Ca sử dụng được xác định khi người dùng xử lý dữ liệu bằng phương pháp cải tiến RXD và truy cập từ bộ công cụ thông qua thanh điều hướng. Luồng cơ bản mô tả trường hợp có thể xảy ra phổ biến nhất trong một ca sử dụng. Luồng cơ bản của ca sử dụng được mô tả như sau:

* + Người dùng truy cập mô đun xử lý dữ liệu từ thanh điều hướng
  + Máy chủ xác nhận quyền sử dụng mô đun
  + Xác nhận quyền thành công
  + Hiển thị giao diện mô đun
  + Chọn dữ liệu đầu vào từ bộ lưu trữ
  + Máy chủ gọi chức năng hiển thị và truy nhập cây thư mục
  + Người dùng tìm và chọn tệp tin dữ liệu đầu vào
  + Máy chủ nhận thông tin tệp dữ liệu
  + Kiểm tra tệp dữ liệu và yêu cầu đầu vào của thuật toán
  + Xác nhận đầu vào hợp lệ
  + Người dùng chọn thuật toán được cài đặt trong module
  + Máy chủ kiểm tra các tham số của thuật toán
  + Hiển thị giao diện nhập tham số
  + Người dùng tùy chỉnh tham số của hệ thống
  + Máy chủ kiểm tra sự phù hợp của tham số
  + Tham số được chấp nhận
  + Người dùng bắt đầu chạy xử lý
  + Máy chủ kiểm tra tài nguyên
  + Thông báo tài nguyên sẵn sang
  + Máy chủ khởi tạo tiến trình xử lý trên máy chủ
  + Máy chủ cập nhật tiến độ xử lý và cập nhật nhật ký xử lý
  + Hiển thị tiến độ xử lý cho người dùng
  + Tiến trình xử lý hoàn tất
  + Máy chủ kiểm tra kết quả xử lý
  + Xử lý thành công
  + Lưu trữ thông tin kết quả
  + Thông báo cho người dùng và hiển thị điều hướng đến trình quản lý dữ liệu

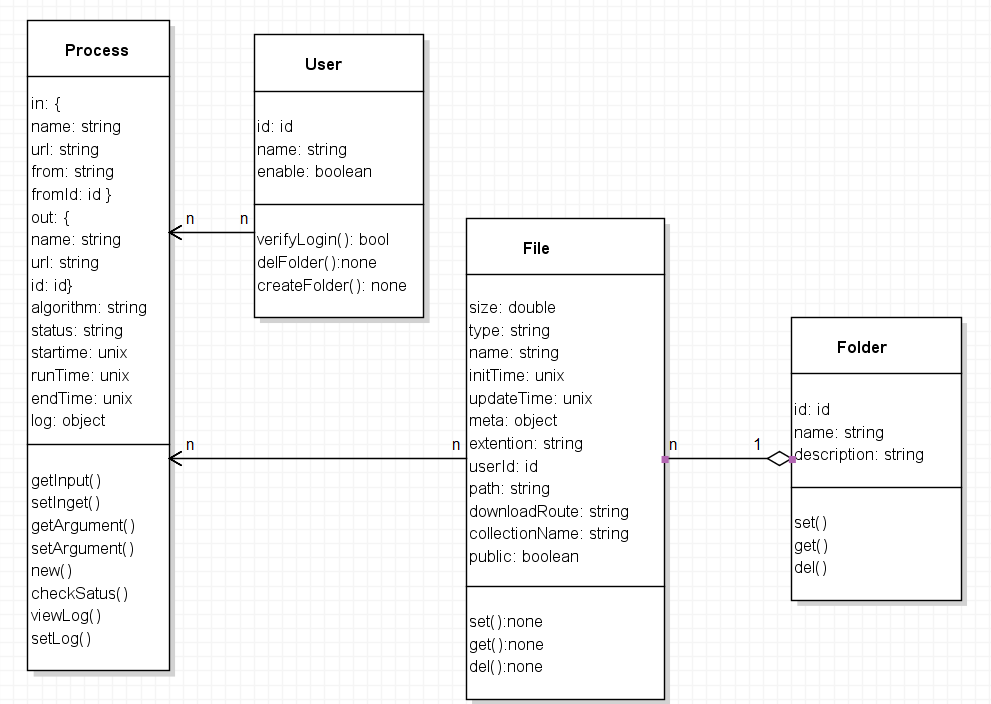
### Luồng rẽ nhánh

Luồng rẽ nhánh mô tả các trường hợp có thể xảy ra trong một lần người dùng sử dụng mô-đun ngoài luồng cơ bản. Công việc phân tích, dự đoán các luồng xử lý của mô-đun nhằm đảm bảo phần mềm có khả năng làm việc trong nhiều trường hợp, giúp hạn chế lỗi trong quá trình sử dụng. Các luồng rẽ nhánh của Mô-đun sử dụng phương pháp cải tiến RXD bao gồm:

* Kết nối máy chủ xử lý bộ công cụ thất bại
  + Người dùng yêu cầu truy cập và yêu cầu được gửi lên máy chủ
  + Yêu cầu được gửi và thời gian chờ phản hồi kết thúc
  + Tập lệnh trạng thái HTTP thông báo cho người dùng kết nối thất bại
* Không tìm thấy dữ liệu đầu vào
  + Người dùng truy cập mô đun xử lý dữ liệu từ thanh điều hướng
  + Máy chủ xác nhận quyền xử dụng mô đun
  + Xác nhận quyền thành công
  + Hiển thị giao diện mô đun
  + Chọn dữ liệu đầu vào từ bộ lưu trữ
  + Máy chủ gọi chức năng hiển thị và truy nhập cây thư mục
  + Người dùng tìm và chọn tệp tin dữ liệu đầu vào
  + Máy chủ nhận thông tin tệp dữ liệu
  + Không tìm thấy dữ liệu được lưu trữ
  + Thông báo lỗi và điều hướng tới chức năng đăng dữ liệu
* Xác nhận dữ liệu đầu vào thất bại
  + Người dùng truy cập mô đun xử lý dữ liệu từ thanh điều hướng
  + Máy chủ xác nhận quyền xử dụng mô đun
  + Xác nhận quyền thành công
  + Hiển thị giao diện mô đun
  + Chọn dữ liệu đầu vào từ bộ lưu trữ
  + Máy chủ gọi chức năng hiển thị và truy nhập cây thư mục
  + Người dùng tìm và chọn tệp tin dữ liệu đầu vào
  + Máy chủ nhận thông tin tệp dữ liệu
  + Kiểm tra tệp dữ liệu và yêu cầu đầu vào của thuật toán
  + Xác nhận đầu vào thất bại
  + Yêu cầu chọn lại dữ liệu đầu vào
* Xác nhận quyền thất bại
  + Người dùng truy cập mô đun xử lý dữ liệu từ thanh điều hướng
  + Máy chủ xác nhận quyền xử dụng mô đun
  + Thông báo không thể truy cập

## Xây dựng biểu đồ lớp

Dựa trên các thông tin đặc tả về ca sử dụng, biểu đồ lớp được xây dựng để mô tả các đối lượng, các lớp đối tượng trong mô-đun và liên kết tương tác giữa các đối tượng đó. Theo tài liệu thiết kế xây dựng bộ công cụ phân tích và xử lý dữ liệu ảnh quang học và radar, cơ sở dữ liệu không quan hệ được sử dụng để lưu trữ các thông tin về quá trình tương tác và xử lý của mô-đun. Do đó, biểu đồ lớp sẽ được sử dụng để quyết định và mô tả cấu trúc lưu trữ trong cơ sở dữ liệu của Mô-đun sử dụng phương pháp cải tiến RXD.



Hình 2.3 Biểu đồ lớp của ca sử dụng

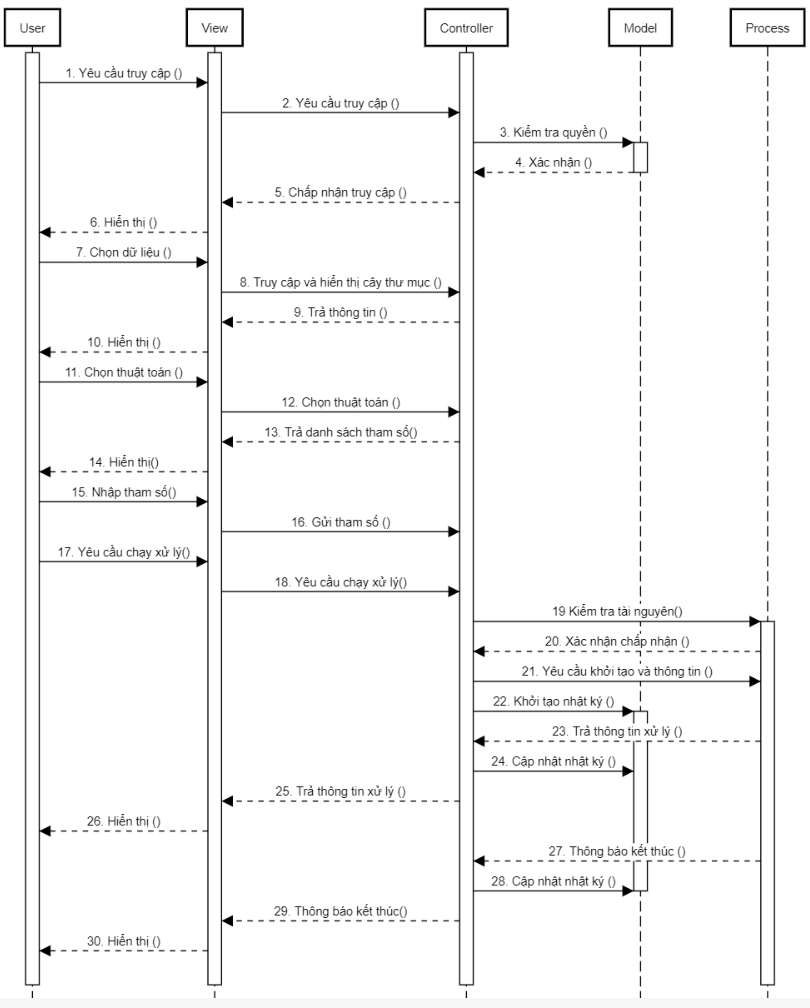
Dựa trên mô tả ca sử dụng, ca sử dụng truy cập bộ công cụ được xác định gồm các lớp:

Bảng 2.1: Các lớp trong ca sử dụng

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Lớp** | **Thuộc tính** | **Mô tả** |
| 1 | Folder | * + Id: id   + Name: string   + Description: string | Lưu trữ vị trí của dữ liệu |
| 2 | File | * size: double * type: string * name: string * initTime: unix * updateTime: unix * meta: object * extention: string * userID: ID * path: string * downloadRoute: string * collectionName: string * public: boolean | Lưu trữ thông tin dữ liệu |
| 3 | Process | * Id: id * In: object   + Name: string   + url: string   + from: string   + fromId: id * out: object   + name: string   + url: string   + id: id * algorithm: string * status: string * startTime: unix * runtime: unix * endTime: unix * log: object | Lưu trữ thông tin tiến trình được thực hiện |

## Xây dựng biểu đồ tuần tự

Biểu đồ trình tự được sử dụng để mô hình hóa tương tác giữa các tác nhân và đối tượng trong phần mềm. Trong Bộ công cụ phân tích và xử lý dữ liệu ảnh quang học và radar, các tương tác của từng mô-đun thể hiện cho các chức năng cần triển khai cài đặt. Dựa trên các đặc tả và các biểu đồ đã phân tích, Hình 2.4 được xây dựng là biểu đồ tuần tự thể hiện trình tự các thao tác của người dùng và tác vụ xử lý của Mô-đun sử dụng phương cải tiến RXD.



Hình 2.4 Biểu đồ tuần tự của ca sử dụng

## Phân tích thành phần Mã nguồn xử lý thuật toán

Spark là một nền tảng tập trung vào khả năng tính toán phân tán và được đánh giá có tốc độ xử lý nhanh với dữ liệu lớn nhờ khả năng phân tán dữ liệu trên bộ nhớ trong thay vì trên bộ nhớ ngoài như Hadoop. Với kích thước hàng triệu điểm ảnh của mỗi tấm ảnh vệ tinh quá trình tính toán và xử lý sẽ được diễn ra nhanh hơn và hiệu quả hơn khi được tính toán trên một cụm phân tán thay vì được tính toán trên một máy tính đơn lẻ. Do các lợi điểm trên, trong Mô-đun sử dụng phương pháp cải tiến RXD, thành phần Mã nguồn xử lý sẽ được thử nghiệm triển khai trên nền tảng Spark.

Phương pháp cải tiến RXD đã trình bày ở Chương 2 có nhiều điểm cải tiến trong cách tiếp cận và công thức. Tuy nhiên, mô hình triển khai của phương pháp này tương đồng với phương pháp RXD đã trình bày trong chuyên đề Xây dựng mô đun sử dụng phương pháp RXD.

# XÂY DỰNG MÔ-ĐUN SỬ DỤNG PHƯƠNG PHÁP CẢI TIẾN RXD

## Trình diễn mã nguồn

### Mã nguồn xử lý thuật toán

Mã nguồn xử lý bao gồm một tập tin mã nguồn sau:

1. **package** build
3. **import** javax.swing.JFrame
5. **import** org.bytedeco.javacpp.opencv\_imgcodecs.\_
6. **import** org.bytedeco.javacv.{CanvasFrame, OpenCVFrameConverter}
7. **import** com.codahale.metrics.Histogram
8. **import** geotrellis.raster.ArrayTile
9. **import** geotrellis.raster.io.geotiff.\_
10. **import** org.bytedeco.javacpp.opencv\_core
12. **import** scala.util.control.Breaks.\_
13. **import** scala.collection.mutable.ArrayBuffer
15. object main{
16. def seaSurfaceAnalysis(numberArr:Array[Short], row:Int, col:Int, size:Int):Array[Float]= {
17. // float: 32bit, double: 64 bit
18. println("start process")
19. val threshold\_1:Int = (0.9 \* size).toInt
20. val threshold\_2:Int = (0.99 \* size).toInt
21. val sizeWin:Int = 5  // 5x5 giong code python | trong luan van 5x5
23. //        Sea surface analysis
24. //        Histogram sorted
25. val histogramArr:Seq[(Short, Int)] = calHistogram(numberArr)
26. println(histogramArr)
28. //      Intensity Discrimination Degree = Majority Intensity Number / Effective Intensity Number
29. var m:Int = 0 // cm at 0.90
30. var e:Int = 0 // ce at 0.99
31. var totalPixel:Int = 0
32. **while** (totalPixel <= threshold\_2) {
33. **if** (totalPixel <= threshold\_1) {
34. m += 1
35. }
36. totalPixel += histogramArr(e).\_2
37. e += 1
38. }
39. println("cm: " + m + ", ce: " + e)
40. val cd:Float = (m \* 1.0f) / e
41. println("cd: " + cd)
43. val histogramMap:Map[Short, Int] = histogramArr.toMap
45. val result:ArrayBuffer[Float] = **new** ArrayBuffer[Float]()
46. **for** (i <- 0 until size) {
47. result.append(
48. (size \* 1.0f / histogramMap(numberArr(i))) \* cd + filter(numberArr, sizeWin, row, col, i) \* (1 - cd)
49. )
50. //            println("complete: " + i\*100.0/size + "%")
51. }
53. result.toArray
54. }
56. def filter(numberArr: Array[Short], sizeWin: Int, row: Int, col: Int, index:Int):Float= {
57. val lenArr:Int = col \* row
58. val win:Int = sizeWin / 2   // 2
59. val run:Int = win \* 2 + 1   // 5
60. val startWin:Int = index - col \* win - win
61. var sum:Int = 0
62. var len:Int = 0
64. val miniWin:ArrayBuffer[Int] = **new** ArrayBuffer[Int]()
65. **for**( i <- 0 until run){ //row
66. **for**( j <- 0 until run){ //col
67. val current:Int = startWin + i \* col + j
68. **if** (current < lenArr && current >= 0) {
69. val nowValue = numberArr(current)
70. sum += nowValue
71. len += 1
72. miniWin.append(nowValue)
73. }
74. }
75. }
76. val avg:Float = sum \* 1.0f / len
77. var sumDiffsSquared:Float = 0
78. **for** (now <- miniWin) {
79. sumDiffsSquared += math.pow(now - avg, 2).toFloat
80. }
81. // return: standard deviation / avg
82. (Math.sqrt(sumDiffsSquared / (len-1)) / avg).toFloat
83. }
85. def calHistogram (numberArr:Array[Short]):Seq[(Short, Int)]= {
86. numberArr
87. .toSeq
88. .groupBy(identity)
89. .mapValues(\_.size)
90. .toSeq
91. .sortBy(- \_.\_2)   //sap xep giam dan theo so luong diem anh
92. }
94. def thresholding (numberArr:Array[Float], row:Int, col:Int, size:Int):Array[Byte]= {
95. val thresholdSet:Int = Math.round(size \* 0.06).toInt
96. var total:Int = 0
97. var threshold:Float = 0
98. val RXDRoad:Array[Float] = numberArr.map(
99. BigDecimal(\_).setScale(3, BigDecimal.RoundingMode.HALF\_UP).toFloat
100. )
101. val histogramRXDRoad: Seq[(Float, Int)] = RXDRoad
102. .toSeq
103. .groupBy(identity)
104. .mapValues(\_.size)
105. .toSeq
106. .sortBy(- \_.\_1)
107. println(histogramRXDRoad)
109. **for**(a <- histogramRXDRoad.indices **if** total < thresholdSet){
110. threshold  = histogramRXDRoad(a).\_1
111. total += histogramRXDRoad(a).\_2
112. }
113. println(threshold)
115. val thresholding:Array[Byte] = RXDRoad.map(road => {
116. **if** (road >= threshold) 4000.toByte
117. **else** 0.toByte
118. })
119. thresholding
120. }
122. //   startup build 102 is on going
123. def main(args: Array[String]): Unit = {
124. println("init")
125. //      get path to build file
126. //        val path = getClass.getResource("").getPath
127. val url: String = "F:\\Code\\sip5\_distributed\\data\\VNREDSAT\_1\\SIP100\\data\\IMAGERY.TIF"
128. val geoTiffSing: SinglebandGeoTiff = SinglebandGeoTiff.streaming(url)
129. println("1. " + geoTiffSing)
130. println("2. " + geoTiffSing.crs)
131. println("3. " + geoTiffSing.cellType)
132. println("4. " + geoTiffSing.extent)
133. println("5. " + geoTiffSing.tags)
134. println("6. " + geoTiffSing.tile)
135. println("7. " + geoTiffSing.raster)
136. println("8. " + geoTiffSing.rasterExtent)
137. println("9. " + geoTiffSing.raster.cols)
138. println("10. " + geoTiffSing.raster.rows)
139. println("11. " + geoTiffSing.raster.size)
141. //        val array: Array[Short] = geoTiffSing.raster.toArray().map(\_.toShort)
142. ////      check lai
143. ////        val array: Array[Short] = geoTiffSing.tile.toArray().map(\_.toShort)
144. //        println("length: " + array.length)
145. //
146. //        val sea:Array[Float] = seaSurfaceAnalysis(array, geoTiffSing.raster.rows, geoTiffSing.raster.cols, geoTiffSing.raster.size)
147. //        val outputdata:ArrayTile = ArrayTile(sea, geoTiffSing.raster.cols, geoTiffSing.raster.rows)
148. //        val output:SinglebandGeoTiff = SinglebandGeoTiff.apply(outputdata, geoTiffSing.extent, geoTiffSing.crs)
149. //        output.write("F:\\Code\\sip5\_distributed\\data\\VNREDSAT\_1\\SIP100\\data\\result\\rxd\_sing.TIF")
150. //
151. ////        build 10
152. ////        var i:Int = 0
153. ////        //remove after version 1.0
154. ////        array.foreach(fileName => {
155. //////            read file by line
156. ////            val currentTextFile:Array[String] = Source
157. ////              .fromFile(url + fileName)
158. ////              .getLines()
159. ////              .toArray
160. ////            val numberArr:Array[Short] = currentTextFile
161. ////              .flatMap(\_.split(","))
162. ////              .map(\_.toShort)
163. //////            count row, col
164. ////            val row:Int = currentTextFile.length
165. ////            val col:Int = numberArr.length / row
166. ////            println("Sizeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee" + "row:" + row + "col:" + col)
167. ////            val writer = new PrintWriter(new File(url + "result" + i ))
168. ////            writer.write(processAllData(numberArr, row,col).mkString("\n"))
169. ////            writer.close()
170. ////            i += 1
171. ////        })
172. //        val threshold:SinglebandGeoTiff = SinglebandGeoTiff.apply(
173. //            ArrayTile(
174. //                thresholding(
175. //                    sea, geoTiffSing.raster.rows, geoTiffSing.raster.cols, geoTiffSing.raster.size
176. //                ), geoTiffSing.raster.cols, geoTiffSing.raster.rows
177. //            ), geoTiffSing.extent, geoTiffSing.crs)
178. //        threshold.write("F:\\Code\\sip5\_distributed\\data\\VNREDSAT\_1\\SIP100\\data\\result\\threshold\_sing.TIF")
179. }
180. }

### Mã nguồn Thành phần thực thi

Mã nguồn Thành phần thực thi của Mô-đun sử dụng phương pháp cải tiến RXD được xây dựng tương ứng với mã nguồn xử lý thuật toán. Mã nguồn chi tiết được trình diễn như sau:

1. **import** { getProcessNameIDByName } from '../config'
2. **const** PROCESS\_NAME = `RXD\_ADVANCED`
3. **const** PROCESS\_ID = getProcessNameIDByName(PROCESS\_NAME)
5. **import** {
6. createPreviewImage,
7. createGDALPreview,
8. createRandomTempFolder,
9. deleteFolder,
10. doLog,
11. excuteSpark,
12. executeCommand,
13. generateParameters,
14. getHDFS,
15. handleProcess,
16. previewName,
17. putHDFS,
18. removeHDFS,
19. hdfs,
20. newLog,
21. endLog
22. } from './executeFunctions'
24. module.exports = {
25. rxdAdvanced
26. }
28. /\*\*
29. \* Phân tích bề mặt biển VNREDSat-1 (rxdAdvanced)
30. \* @param {Object} input
31. \* @return {Promise}
32. \*/
33. function rxdAdvanced(input) {
34. let logID
35. **return** **new** Promise((resolve, reject) => {
36. //Ex: /user/hadoop/VNREDSat-1/VNR20150415\_PAN.tif
37. **const** inImageHDFSPath = input.inputImage.split(`/`)
39. //Lấy thư mục tổng của ảnh VNREDSat-1
40. let rootImageFolderTemp = input.inputImage.split(`/`)
41. rootImageFolderTemp.length = rootImageFolderTemp.length - 1
42. **const** rootImageFolder = rootImageFolderTemp.join(`/`).substr(1).concat('/') // --> user/hadoop/
43. **const** imageName = inImageHDFSPath[inImageHDFSPath.length - 1] //-->VNR20150415\_PAN.tif,
44. **const** imageNameNoExtent = imageName.split(`.`)[0] //-->VNR20150415\_PAN
46. **const** moduleParam = **new** Map();
47. moduleParam.set(`--**class**`, `pack.rxd`)
48. moduleParam.set(`--master`, `yarn`)
49. moduleParam.set(`--deploy-mode`, `cluster`);
50. moduleParam.set(`--total-executor-cores`, `4`);
51. moduleParam.set(`--driver-memory`, `6g`);
52. moduleParam.set(`--executor-memory`, `6g`);
53. moduleParam.set(`--conf`, `spark.kryoserializer.buffer.max=128m`);
54. **const** rxdParams = generateParameters(moduleParam)
55. rxdParams.push(`hdfs://${hdfs}user/hadoop/code/sip5\_distributed-1.6.5\_RXD.jar`)
56. rxdParams.push(hdfs)
57. rxdParams.push(rootImageFolder)
58. rxdParams.push(imageNameNoExtent)
60. newLog(PROCESS\_ID, input.userID || Meteor.userId() || ``, input).then(id => {
61. logID = id
62. **return** excuteSpark(rxdParams, logID)
63. }).then(() => {
64. endLog(logID)
65. resolve()
66. }).**catch**(err => {
67. doLog(err, logID)
68. endLog(logID)
69. reject()
70. })
71. })
72. }

### Mã nguồn giao diện tương tác

Mã nguồn giao diện tương tác gồm 2 phần:

* Mã nguồn quản lý tương tác, thay đổi và cung cấp các chức năng của giao diện

1. **import** { Template } from 'meteor/templating'
2. **import** { Meteor } from 'meteor/meteor'
3. **import** {
4. getProcessNameIDByName,
5. getProcessTitleByID
6. } from '../../../../../api/sip9/toolbox/config'
7. **import** { Session } from 'meteor/session'

10. **import** {
11. checkRequired,
12. genSelectFile,
13. bindSelectFile,
14. getFile,
15. toastTime,
16. initMaterialize,
17. getInput,
18. checkRunning
19. } from '../../../../helpers/hue'
21. **import** './rxdAdvanced.html';
23. **const** PROCESS\_NAME = `RXD\_ADVANCED`
24. **const** PROCESS\_ID = getProcessNameIDByName(PROCESS\_NAME)
25. **const** PROCESS\_TITLE = getProcessTitleByID(PROCESS\_ID)
27. //Sửa các extension khung chọn file có thể chọn
28. **const** ACCEPT\_FILE\_TYPE = [`tif`, 'tiff', `TIF`, 'TIFF']
30. Template.rxdAdvanced.onCreated(**function** () {
31. Session.set('PROCESS\_ID', PROCESS\_ID)
32. Session.set('ACCEPT\_FILE\_TYPE', ACCEPT\_FILE\_TYPE)
33. })
35. Template.rxdAdvanced.events({
36. //Nút chọn ảnh --> ko sửa
37. 'click #chose': () => {
38. genSelectFile(ACCEPT\_FILE\_TYPE, `inputImage`);
39. },
40. 'click #submit': () => {
41. **if**(checkRunning())
42. Materialize.toast(`Đang có tiến trình thực thi`, toastTime);
43. **else** **if** (checkRequired()) {
44. Materialize.toast(`Bat dau xu ly`, toastTime);
45. let input = {
46. inputImage: $('#inputImage').val()
47. }
49. Meteor.call(`rxdAdvanced`, input, (err, result) => {
50. **if** (err) {
51. console.log(err)
52. } **else** {
53. console.log(`done`)
54. }
55. })
56. } **else** {
57. Materialize.toast(`Vui lòng điền hết các trường bắt buộc`, toastTime);
58. }
59. }
60. });
62. Template.rxdAdvanced.onRendered(() => {
63. initMaterialize()
64. **if**(getInput()){
65. Meteor.call(`getLog`, getInput(), (err, input) => {
66. **if** (err || !input) {
67. console.log(err)
68. } **else** {
69. $('#inputImage').val(input.inputImage)
70. }
71. })
72. }
73. });

* Mã nguồn giao diện

1. <template name="rxdAdvanced">
2. <h2>{{ processTitle }}</h2>
3. <div **class**="row">
4. <div **class**="row">
5. <div **class**="input-field col s10">
6. <input id="inputImage" type="text" placeholder="Ảnh đầu vào" **class**="validate" disabled required>
7. </div>
8. <div **class**="input-field col s2">
9. <button **class**="btn waves-effect waves-light" id="chose">Chọn
10. </button>
11. </div>
12. </div>
13. <h4 **class**="header">Mô đun sử dụng phương pháp RXD cai tien</h4>
14. <p>! Chức năng không yêu cầu tham số đặc biệt</p>
16. <button **class**="btn waves-effect waves-light" id="submit" name="action">Submit</button>
17. </div>
18. </template>

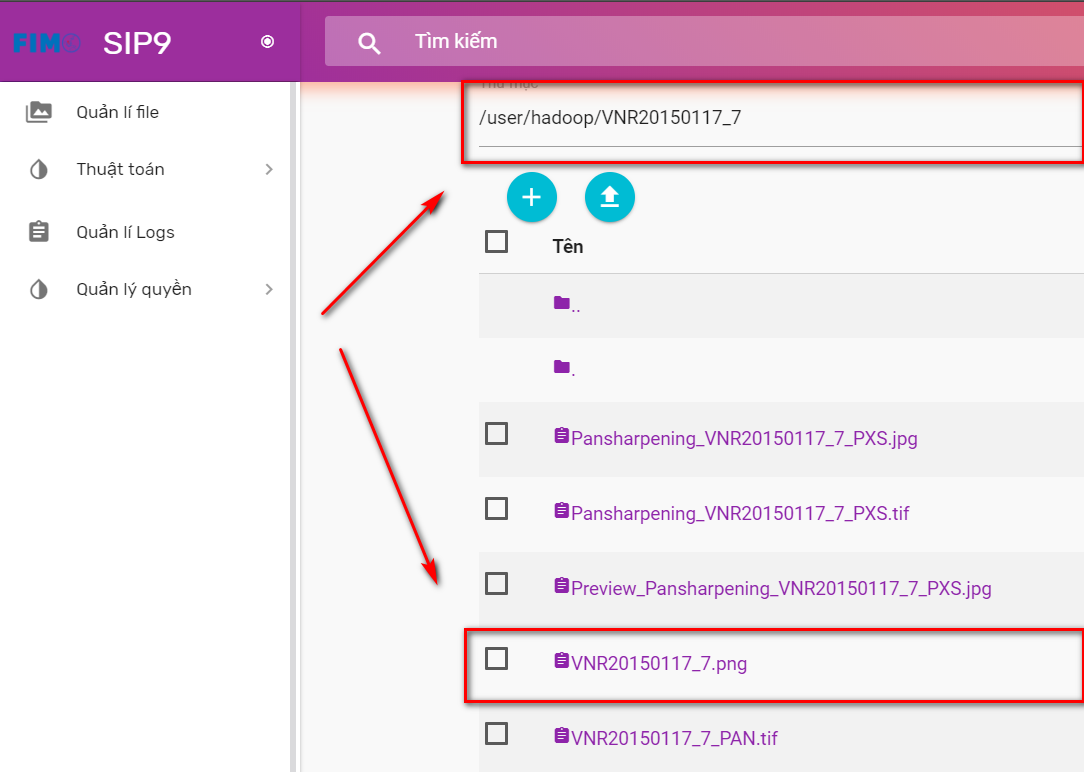
## Trình diễn mô-đun

Mô-đun sử dụng phương pháp cải tiến RXD đã được thử nghiệm với dữ liệu ảnh:

* + Tên ảnh: VNR20150117
  + Loại ảnh: VNREDSat-1
  + Vị trí chụp: đảo Chữ Thập
  + Thời gian chụp: 17/1/2015

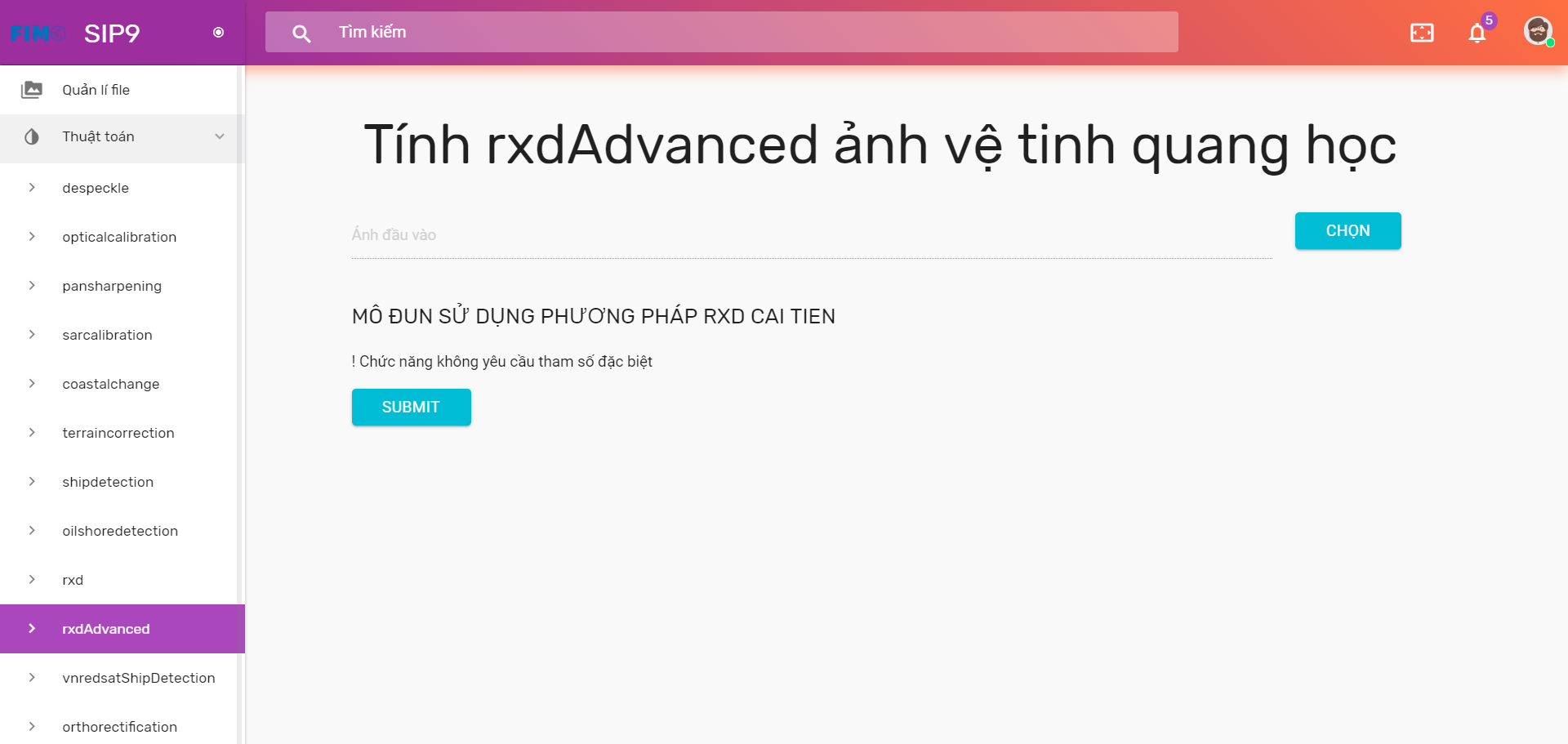
Các thao tác để sử dụng Mô-đun sử dụng phương pháp cải tiến RXD được mô tả như sau:

* + Truy cập phần quản lý dữ liệu và tải ảnh vệ tinh lên bộ công cụ



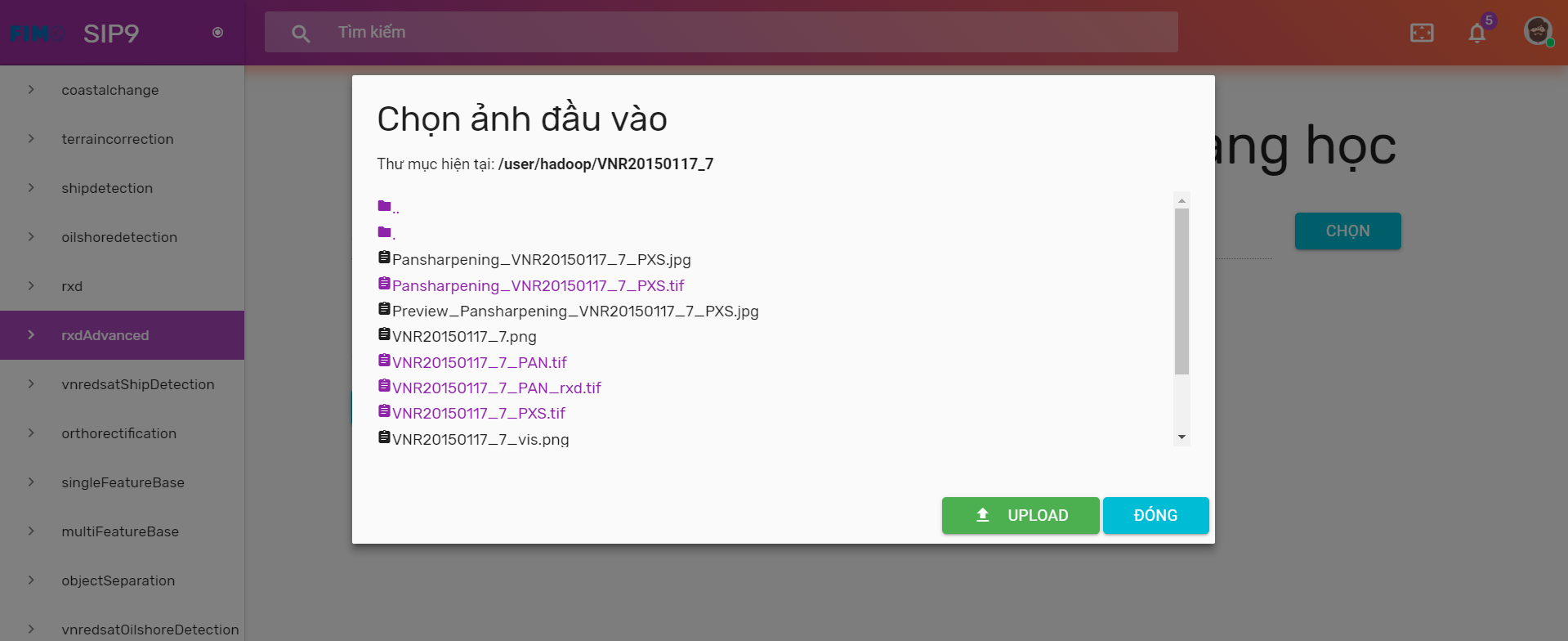
Hình 3.1: quản lý dữ liệu và tải ảnh vệ tinh lên bộ công cụ

* + Truy cập Mô-đun sử dụng phương pháp cải tiến RXD

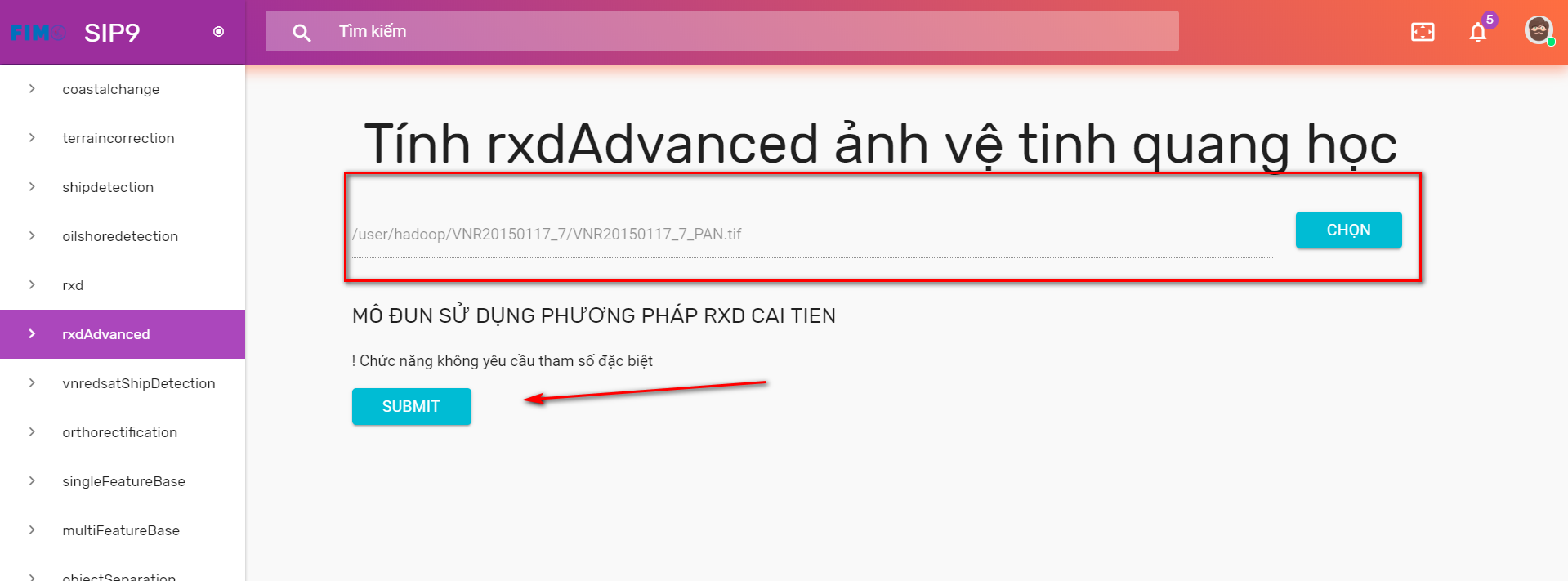


Hình 3.2: Mô-đun sử dụng phương pháp RXD cải tiến

* Chọn ảnh đầu vào để tiến hành chạy Mô-đun sử dụng phương pháp cải tiến RXD

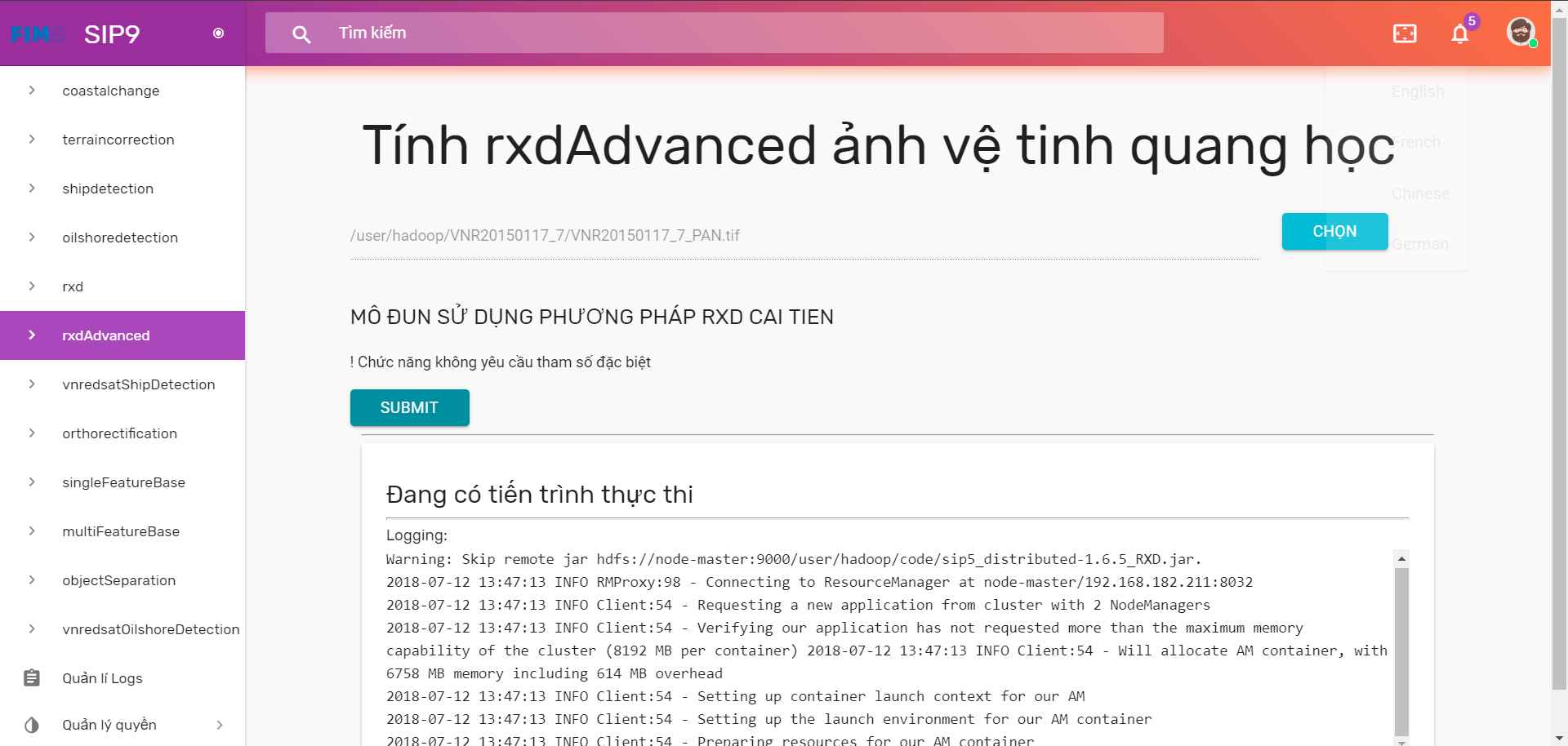


Hình 3.3: Lựa chọn ảnh đầu vào



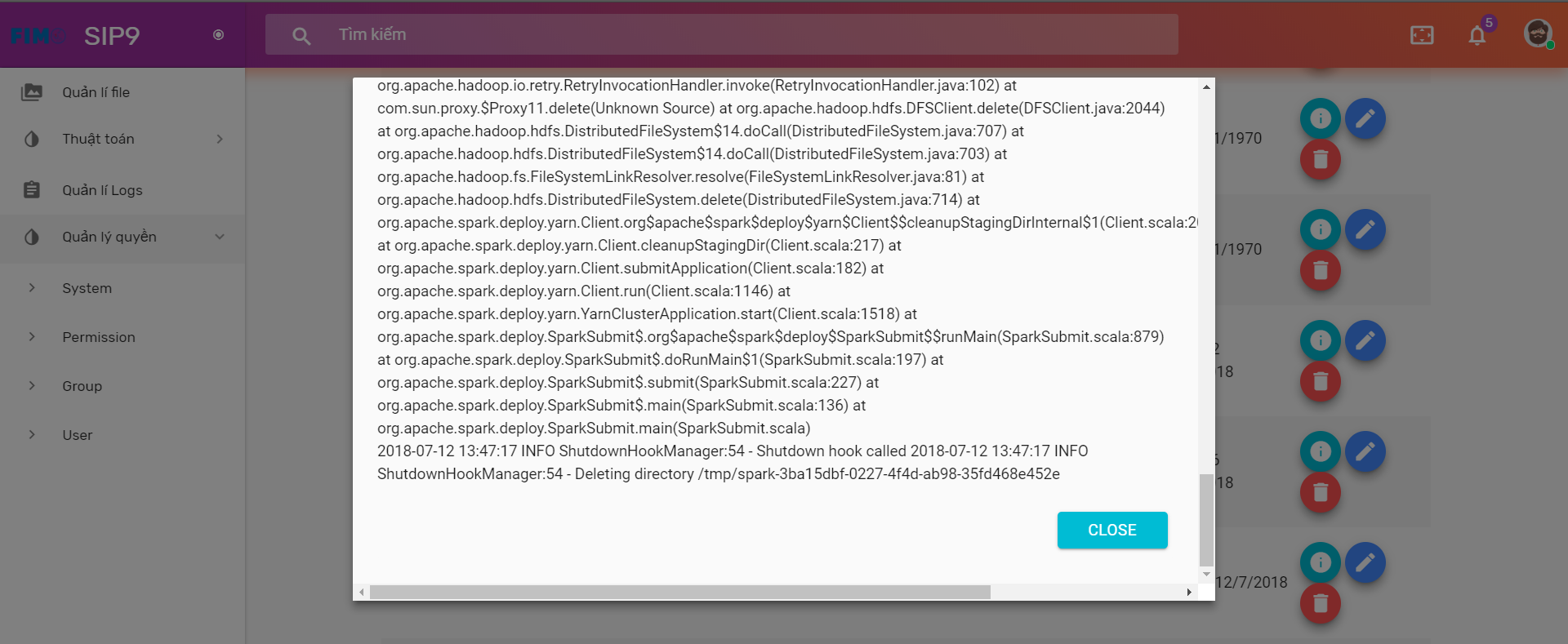
Hình 3.4: Mô-đun đã nhận dữ liệu ảnh đầu vào

Sau khi chọn xong dữ liệu, người dùng chọn submit để tiến hành chạy thuật toán tính RXD cải tiến cho ảnh vệ tinh quang học. Khi chạy Mô-đun sử dụng phương pháp cải tiến RXD, nhật ký của chương trình sẽ được hiển thị phía bên dưới giao diện thuật toán, và người dùng sẽ được thông báo rằng tiến trình xử lý thuật toán đang được thực thi như trong Hình 3.5



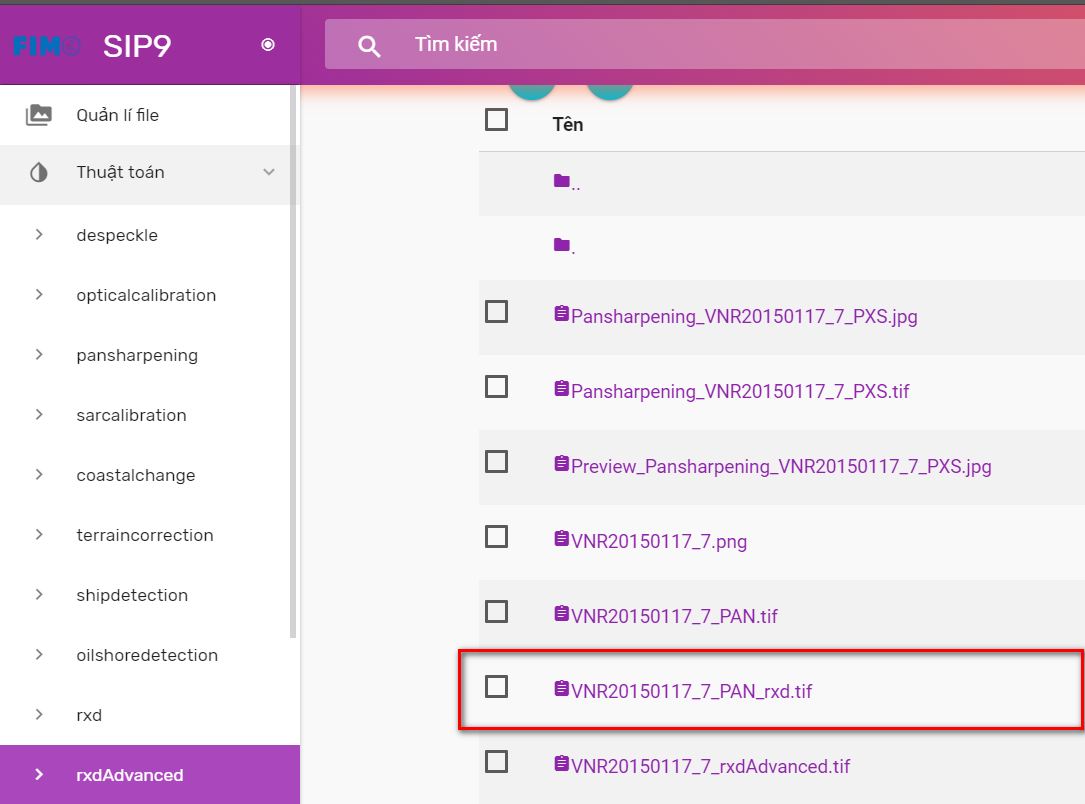
Hình 3.5: Quá trình xử lý đang diễn ra

Người sử dụng có thể xem logs chương trình trong quá trình chạy xử lý mô-đun thông qua phần quản lý tiến trình và log như trong Hình 3.6



Hình 3.6: Nhật ký thực thi chương trình

Sau khi chạy xử lý xong, dữ liệu kết quả sẽ được lưu cùng thư mục với ảnh dữ liệu đầu vào. Ảnh kết quả sẽ có tên là tên của ảnh ban đầu cộng thêm chuỗi kí tự “\_RXDAdvanced” như trong Hình 3.7



Hình 3.7: Kết quả xử lý

Khi quá trình xử lý đã hoàn tất, người dùng có thể xem toàn bộ nhật ký xử lý dữ liệu. Giao diện của chức năng này được thể hiện trong trình quản lý nhật ký xử lý như trong Hình 3.8



Hình 3.8: Quản lý lịch sử của mô-đun

Người dùng có thể chọn các thao tác đã được tích hợp sẵn vào trong hệ thống để hủy tiến trình hoặc thay đổi tiến trình và xem đầy đủ các nhật ký của hệ thống.

# KẾT LUẬN

Trong chuyên đề này, mô đun sử dụng phương pháp cải tiến RXD đã được trình bày, cụ thể bao gồm phương pháp xử lý, phân tích thiết kế và xây dựng mô đun. Mô-đun này cung cấp khả năng xác định các đối tượng dị thường trên bề mặt biển bằng việc xác định sự khác biệt phổ và giá trị giữa vùng cần kiểm tra và các điểm lân cận của nó hoặc toàn bộ ảnh. Việc xây dựng mô đun này đã cải mức độ hiệu quả và cung cấp phương pháp giúp thuận lợi hơn trong việc xác định đối tượng dị thường trên bề mặt biển.

Xây dựng Mô đun sử dụng phương pháp cải tiến RXD đã nghiên cứu lý thuyết cải tiến phương pháp, phân tích thiết kế chi tiết các thành phần của mô-đun. Từ đó tiến hành triển khai phương pháp xử lý, cách thức tương tác, giao diện diện và chức năng. Kết quả của quá trình triển khai là mã nguồn hoàn thiện của các thành phần và mô-đun đã được thử nghiệm trên dữ liệu ảnh VNREDSat-1.

|  |  |
| --- | --- |
| **Chủ nhiệm đề tài**  *(ký và ghi rõ họ và tên)* | **Thủ trưởng tổ chức chủ trì đề tài**  *(ký tên và đóng dấu)* |

# TÀI LIỆU THAM KHẢO

[1] Burns, J. B., Hanson, A. R., & Riseman, E. M. (1986). Extracting Straight Lines. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-8(4), 425–455. https://doi.org/10.1109/TPAMI.1986.4767808.

[2] Crisp, D. J. (2004). The State-of-the-Art in Ship Detection in Synthetic Aperture Radar Imagery.

[3] Zou, Z.; Shi, Z. Ship Detection in Spaceborne Optical Image With SVD Networks. *IEEE Trans. Geosci. Remote Sens.* **2016**, *54*, 5832–5845, doi:10.1109/TGRS.2016.2572736.

[4] Desolneux, A., Ladjal, S., Moisan, L., & Morel, J. M. (2002). Dequantizing image orientation. IEEE Transactions on Image Processing, 11(10), 1129–1140. https://doi.org/10.1109/TIP.2002.804566.

[5] Desolneux, A., Moisan, L., & Morel, J. M. (2001). Edge detection by Helmholtz principle. Journal of Mathematical Imaging and Vision, 14(3), 271–284. https://doi.org/10.1023/A:1011290230196.

[6] Grompone von Gioi, R., Jakubowicz, J., Morel, J.-M., & Randall, G. (2012). LSD: a Line Segment Detector. Image Processing On Line, 2, 35–55. https://doi.org/10.5201/ipol.2012.gjmr-lsd.

[7] Guang Yang, Bo Li, Shufan Ji, Feng Gao, & Qizhi Xu. (2014). Ship Detection From Optical Satellite Images Based on Sea Surface Analysis. IEEE Geoscience and Remote Sensing Letters, 11(3), 641–645. https://doi.org/10.1109/LGRS.2013.2273552.

[8] Luu, V. H., Pham, V. M., Man, D. C., Bui, Q. H., & Nguyen, T. N. T. (2016). Comparison of various image fusion methods for impervious surface classification from VNREDSat-1. International Journal of Advanced Culture Technology (IJACT), 4(2), 1–6.

[9] Rodarmel, C., & Shan, J. (2002). Principal component analysis for hyperspectral image classification. Surveying and Land Information Science, 62(2), 115. https://doi.org/10.1109/IGARSS.2001.976068.

[10] von Gioi, R. G., Jakubowicz, J., Morel, J.-M., & Randall, G. (2010). LSD: A Fast Line Segment Detector with a False Detection Control. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(4), 722–732. https://doi.org/10.1109/TPAMI.2008.300.

[11] Zhang, Y. (n.d.). Highlight Article: Understanding Image Fusion. Photogrammetric Engineering & Remote Sensing, 70(6), 657–661.

[12] Zhang, Y., & Mishra, R. K. (2013). From UNB PanSharp to Fuze Go – the success behind the pan-sharpening algorithm. International Journal of Image and Data Fusion, 5(00), 39–53. https://doi.org/10.1080/19479832.2013.848475.