**Mở đầu**

1. **Giới thiệu tổng quan đề tài**

**Mô tả bài toán: Hệ thống phân loại độ chín của cà chua dựa trên nhận diện màu sắc**

Trong lĩnh vực nông nghiệp và sản xuất thực phẩm, việc đánh giá độ chín của cà chua là một bước quan trọng để đảm bảo chất lượng và tối ưu hóa chuỗi cung ứng. Hiện tại, quá trình này thường được thực hiện thủ công, dễ dẫn đến sai sót và không đảm bảo tính nhất quán. Vì vậy, việc xây dựng một hệ thống tự động phân loại độ chín của cà chua dựa trên màu sắc sẽ giúp cải thiện độ chính xác và hiệu quả.

**Bài toán cụ thể**

Hệ thống cần thực hiện các nhiệm vụ sau:

1. **Nhận diện màu sắc từ hình ảnh:**
   * Hệ thống sẽ nhận đầu vào là các hình ảnh chụp cà chua.
   * Phân tích hình ảnh để nhận diện màu sắc của từng vùng hoặc toàn bộ cà chua bằng cách sử dụng OpenCV.
2. **Xử lý và phân loại màu sắc:**
   * Sử dụng các mô hình màu (HSV) để xác định các nhóm màu đặc trưng như xanh, vàng, đỏ.
   * Phân tích tỷ lệ các màu sắc chiếm ưu thế trong từng quả cà chua để đánh giá độ chín.
3. **Đánh giá độ chín của cà chua:**
   * Dựa trên các ngưỡng màu sắc đã định nghĩa, phân loại cà chua vào các mức độ chín: chưa chín (xanh), chín vừa (vàng), và chín đỏ hoàn toàn.

**Mục tiêu và ý nghĩa**

Hệ thống này không chỉ giúp tự động hóa quá trình đánh giá độ chín, mà còn cung cấp thông tin chính xác và dễ dàng phân tích để hỗ trợ các quyết định trong:

* **Quản lý sản phẩm nông sản:** Phân loại sản phẩm trước khi đưa vào tiêu thụ hoặc chế biến.
* **Giảm lãng phí:** Giảm thiểu sự sai lệch do con người, đảm bảo chỉ tiêu thụ sản phẩm đúng độ chín.
* **Tối ưu hóa chuỗi cung ứng:** Đưa ra các dữ liệu định lượng để lên kế hoạch vận chuyển và lưu trữ phù hợp.

**Công nghệ áp dụng**

* **OpenCV:** Xử lý và phân tích hình ảnh để nhận diện màu sắc.
* **Pandas:** Lưu trữ và phân tích dữ liệu màu sắc đã nhận diện để tổng hợp thông tin.
* **Python:** Ngôn ngữ lập trình chính để triển khai hệ thống.

1. **Phân công nhiệm vụ**

Dưới đây là bảng phân công nhiệm vụ chi tiết:

|  |  |  |
| --- | --- | --- |
| **Thành viên** | **Nhiệm vụ** | **Mô tả** |
| Trần Khắc Sơn | Chuẩn bị môi trường phát triển | - Cài đặt và kiểm tra các thư viện: OpenCV, Pandas, Matplotlib. - Cấu hình IDE (PyCharm). |
| Trần Hồng Ánh | Phát triển chương trình xử lý ảnh | - Chuyển đổi ảnh sang không gian màu HSV. - Tính toán histogram cho từng kênh màu (H, S, V). |
| Bùi Thị Lê Na | Phân tích kết quả từ histogram   |  | | --- | |  | | - Xác định màu sắc chủ đạo dựa trên đỉnh histogram. - Lưu kết quả vào Pandas. |
| Nguyễn Đức Trung | Xây dựng giao diện hiển thị | - Phát triển giao diện đồ họa đơn giản bằng PyQt. - Hiển thị kết quả phân tích. |
| Vũ Hồng Phong | Kiểm thử và viết báo cáo | - Kiểm thử với các bộ ảnh khác nhau để đánh giá độ chính xác. - Tổng hợp và viết báo cáo. |

**Chương I: Cơ sở lý thuyết**

1. **Các thuật toán**

**1.1. Thuật toán GrabCut**

**a. Khái niệm**

GrabCut là một thuật toán phân đoạn ảnh tiên tiến được giới thiệu vào năm 2004 bởi Rother và cộng sự. Đây là một phương pháp dựa trên lý thuyết đồ thị để tách đối tượng chính (foreground) khỏi nền (background) trong ảnh, sử dụng mô hình thống kê Gaussian Mixture Model (GMM) và tối ưu hóa Graph Cut.

**Nguyên lý hoạt động**

* **Đầu vào:** Người dùng cung cấp một hình chữ nhật bao quanh đối tượng cần tách. Các pixel nằm ngoài hình chữ nhật được gán là nền, trong khi các pixel bên trong thuộc vùng chưa xác định.
* **Xây dựng mô hình Gaussian Mixture Model (GMM):** Pixel trong ảnh được phân nhóm dựa trên các đặc tính màu sắc. GMM sử dụng các phân phối Gaussian để đại diện cho các cụm màu sắc trong cả vùng nền và vùng đối tượng.
* **Tối ưu hóa bằng Graph Cut:**
  + Ảnh được biểu diễn như một đồ thị, nơi mỗi pixel là một nút, và các cạnh giữa các nút thể hiện sự liên kết giữa các pixel (chẳng hạn, độ tương đồng màu sắc).
  + Thuật toán tối ưu hóa năng lượng để tìm ra ranh giới tốt nhất giữa nền và đối tượng.
* **Lặp lại:**
  + Mô hình GMM được cập nhật sau mỗi lần phân đoạn. Thuật toán tiếp tục cho đến khi không có thay đổi lớn trong kết quả.

**b, Ứng dụng trong bài toán**

GrabCut được dùng để tách riêng quả cà chua khỏi nền, giúp quá trình phân tích chỉ tập trung vào vùng chứa thông tin cần thiết, loại bỏ các yếu tố gây nhiễu như ánh sáng, bóng đổ hay các vật thể xung quanh.

**1.2. Không gian màu HSV**

**a. Khái niệm**

Không gian màu HSV (Hue, Saturation, Value) là một hệ tọa độ màu phổ biến trong xử lý ảnh. HSV được thiết kế để phản ánh cách con người nhận thức về màu sắc, giúp dễ dàng phân tích và phân loại các yếu tố liên quan đến màu.

* **Hue (H):** Đại diện cho màu sắc chính, được đo bằng góc từ 0° đến 360°.
* **Saturation (S):** Độ bão hòa, cho biết mức độ tinh khiết của màu, từ 0 (màu xám) đến 255 (màu sắc đậm nhất).
* **Value (V):** Độ sáng, phản ánh mức độ ánh sáng của màu, từ 0 (đen) đến 255 (sáng nhất).

**b. Nguyên lý sử dụng HSV trong bài toán**

* **Hue:** Được sử dụng để xác định màu sắc chủ đạo của quả cà chua. Ví dụ:
  + Màu đỏ hoặc đỏ cam: Quả chín.
  + Màu xanh lá: Quả chưa chín.
* **Saturation:** Phản ánh độ tươi của quả. Quả chín thường có độ bão hòa cao, trong khi quả hỏng có thể có độ bão hòa thấp hơn.
* **Value:** Phát hiện các vùng tối hoặc mờ có thể là dấu hiệu của tổn thương hoặc hỏng hóc.

Việc chuyển đổi ảnh từ không gian màu BGR sang HSV giúp cải thiện khả năng phân tích, vì HSV tách biệt rõ ràng thông tin về màu sắc, độ sáng, và bão hòa, làm cho quá trình nhận diện màu sắc trở nên trực quan hơn.

**1.3. Biểu đồ Histogram**

**a. Khái niệm**  
Histogram là một công cụ phân tích thống kê phổ biến, biểu diễn sự phân bố các giá trị pixel trong một hình ảnh hoặc một vùng cụ thể.

**b. Nguyên lý hoạt động**

* **Thu thập dữ liệu:**
  + Các kênh màu trong không gian HSV (H, S, V) được chia thành các dải giá trị (bins).
  + Histogram đếm số lượng pixel nằm trong mỗi dải.
* **Phân tích phân bố:**
  + Với kênh Hue: Histogram cho thấy các màu sắc xuất hiện trong ảnh và tần suất của chúng.
  + Với kênh Saturation và Value: Histogram giúp xác định độ sáng và độ bão hòa của ảnh.
* **Ứng dụng trong bài toán:**
* **Phân loại độ chín:**  
  Histogram Hue được sử dụng để xác định màu sắc chủ đạo trong ảnh. Dựa trên màu sắc chiếm ưu thế, quả cà chua có thể được phân loại là: chín, hơi xanh, hoặc chưa chín.
* **Phát hiện bất thường:**  
  Sự dao động mạnh hoặc sự xuất hiện của các giá trị ngoài phạm vi bình thường (ví dụ: vết thâm hoặc vết đổi màu) có thể chỉ ra rằng quả cà chua bị hỏng.

**Kết luận**

Trong bài toán nhận diện và phân loại độ chín của cà chua:

* GrabCut giúp tách vùng cà chua khỏi nền để tập trung phân tích.
* HSV cung cấp một công cụ mạnh mẽ để phân tích màu sắc và đặc tính quang học của quả.
* Histogram hỗ trợ trực quan hóa và định lượng các thông tin cần thiết, phục vụ quá trình phân loại và phát hiện bất thường.

1. **Ngôn ngữ và môi trường**

**2.1. Ngôn ngữ lập trình Python**

Python là một ngôn ngữ lập trình bậc cao, được phát triển với mục tiêu dễ học và dễ sử dụng. Nhờ cú pháp đơn giản, rõ ràng và dễ hiểu, Python đã trở thành lựa chọn phổ biến trong nhiều lĩnh vực khác nhau, bao gồm khoa học dữ liệu, trí tuệ nhân tạo, và xử lý ảnh. Với khả năng xử lý các tác vụ phức tạp một cách hiệu quả và các thư viện phong phú hỗ trợ nhiều loại hình công việc, Python đặc biệt phù hợp cho các dự án yêu cầu tính linh hoạt, dễ bảo trì và mở rộng.

Trong dự án này, Python được lựa chọn vì các lý do sau:

* **Đơn giản và dễ duy trì:** Python có cú pháp dễ đọc và dễ hiểu, giúp giảm thiểu thời gian viết mã và bảo trì hệ thống trong suốt quá trình phát triển và vận hành.
* **Khả năng tương thích cao:** Python có thể chạy trên nhiều hệ điều hành khác nhau như Windows, macOS, và Linux, giúp người phát triển có thể triển khai ứng dụng dễ dàng trên nhiều nền tảng mà không cần thay đổi mã nguồn quá nhiều.
* **Khả năng mở rộng và tích hợp:** Python có thể dễ dàng kết hợp với các công nghệ và công cụ khác. Nhờ vào khả năng mở rộng cao, Python có thể hỗ trợ việc tích hợp thêm các công nghệ mới khi dự án phát triển thêm tính năng hoặc yêu cầu thay đổi.

Đặc biệt, Python cũng được hỗ trợ mạnh mẽ trong các ứng dụng khoa học dữ liệu và xử lý ảnh, cho phép các thuật toán phân tích và xử lý ảnh (như GrabCut hay các phương pháp phân loại màu sắc) được triển khai một cách hiệu quả và dễ dàng. Vì vậy, Python là một lựa chọn lý tưởng cho việc xây dựng hệ thống nhận diện màu sắc và phân loại độ chín của cà chua.

**2.2. Môi trường phát triển PyCharm**

PyCharm là một môi trường phát triển tích hợp (IDE) mạnh mẽ, được tối ưu hóa cho việc lập trình với Python. Được phát triển bởi JetBrains, PyCharm cung cấp một loạt các tính năng mạnh mẽ hỗ trợ lập trình viên trong suốt quá trình phát triển phần mềm, từ việc viết mã đến kiểm tra và gỡ lỗi. Với những tính năng ưu việt, PyCharm trở thành một công cụ không thể thiếu trong phát triển phần mềm Python, đặc biệt là trong các dự án yêu cầu tính chính xác và tối ưu cao như xử lý ảnh và phân tích dữ liệu.

Lý do chọn PyCharm cho dự án này bao gồm:

* **Giao diện người dùng thân thiện và dễ sử dụng:** PyCharm cung cấp giao diện dễ nhìn và tổ chức hợp lý, giúp người lập trình dễ dàng quản lý mã nguồn, thư viện, và các tệp liên quan đến dự án.
* **Hỗ trợ gỡ lỗi mạnh mẽ:** PyCharm tích hợp các công cụ gỡ lỗi rất mạnh mẽ, giúp lập trình viên có thể theo dõi từng bước thực thi của mã nguồn và phát hiện lỗi nhanh chóng trong quá trình phát triển. Điều này rất quan trọng khi xử lý các thuật toán phức tạp như GrabCut, nơi cần phải kiểm tra và tinh chỉnh kết quả phân đoạn ảnh.
* **Quản lý môi trường phát triển (Virtual Environments):** PyCharm hỗ trợ việc tạo và quản lý môi trường ảo, giúp dễ dàng duy trì các phiên bản thư viện khác nhau cho từng dự án. Điều này giúp tránh xung đột giữa các phiên bản và tạo môi trường phát triển ổn định và hiệu quả.

Ngoài ra, PyCharm cũng hỗ trợ tích hợp Git để quản lý mã nguồn và hỗ trợ tính năng tự động hoàn thành mã, giúp tăng tốc quá trình phát triển phần mềm. Những tính năng này đảm bảo rằng việc phát triển, kiểm tra và bảo trì hệ thống nhận diện màu sắc và phân loại độ chín của cà chua sẽ được thực hiện nhanh chóng và hiệu quả.

**2.3. Kết hợp Python và PyCharm trong bài toán**

Với sự kết hợp giữa Python và PyCharm, quá trình phát triển hệ thống nhận diện màu sắc và phân loại độ chín của cà chua trở nên dễ dàng và hiệu quả. Python đảm nhiệm các tác vụ xử lý và phân tích dữ liệu, từ việc xử lý hình ảnh với các thuật toán như GrabCut đến phân loại màu sắc và phát hiện độ chín của quả cà chua. Cùng với đó, PyCharm hỗ trợ tối đa trong việc phát triển, kiểm tra và bảo trì mã nguồn, giúp lập trình viên dễ dàng tối ưu hóa và phát triển thêm tính năng cho dự án.

Python và PyCharm kết hợp mang lại một môi trường phát triển mạnh mẽ, linh hoạt và dễ duy trì, giúp dự án có thể mở rộng trong tương lai và đáp ứng các yêu cầu phức tạp trong xử lý ảnh và phân tích dữ liệu.

1. **Module, công cụ và thư viện**

Hệ thống nhận diện màu sắc và phân loại độ chín của cà chua được xây dựng dựa trên sự kết hợp của nhiều module, công cụ và thư viện mạnh mẽ. Những thành phần này không chỉ hỗ trợ về mặt xử lý dữ liệu và hình ảnh, mà còn giúp xây dựng giao diện người dùng trực quan và thân thiện. Dưới đây là mô tả chi tiết từng công cụ, module và thư viện được sử dụng trong dự án.

* 1. **PyQt5**

**PyQt5** là một bộ công cụ GUI mạnh mẽ, cung cấp các công cụ và widget cần thiết để xây dựng các ứng dụng desktop hiện đại. Được xây dựng dựa trên Qt Framework, PyQt5 hỗ trợ đa nền tảng và cho phép tạo ra các ứng dụng có giao diện tương tác cao.

Trong dự án, PyQt5 đóng vai trò trung tâm trong việc xây dựng giao diện người dùng, với các chức năng chính bao gồm:

* **Tạo giao diện chính của ứng dụng:** Giao diện này bao gồm cửa sổ chính (Main Window) và các thành phần giao diện như nút bấm, hộp thoại chọn file (File Dialog), nhãn hiển thị hình ảnh (Label), và vùng hiển thị kết quả.
* **Hỗ trợ sự kiện tương tác:** Các sự kiện như chọn ảnh từ thư mục, khởi chạy phân tích hình ảnh, và hiển thị kết quả được quản lý thông qua PyQt5.
* **Kết nối giữa giao diện và thuật toán:** PyQt5 đảm bảo việc truyền dữ liệu giữa giao diện người dùng và các module xử lý như OpenCV hoặc NumPy diễn ra mượt mà.

Ngoài ra, PyQt5 tích hợp với **Qt Designer**, một công cụ thiết kế giao diện trực quan, giúp giảm thời gian và công sức trong việc tạo bố cục giao diện. Các tệp .ui được tạo ra từ Qt Designer sau đó được chuyển đổi thành mã Python để sử dụng trong dự án.

* 1. **OpenCV**

**OpenCV (Open Source Computer Vision)** là một thư viện mã nguồn mở chuyên về xử lý ảnh và thị giác máy tính. Đây là một trong những công cụ mạnh mẽ và phổ biến nhất hiện nay để xử lý hình ảnh kỹ thuật số.

Trong dự án, OpenCV được ứng dụng để:

* **Xử lý hình ảnh đầu vào:** Đọc file ảnh, chuyển đổi định dạng màu từ BGR sang HSV (Hue, Saturation, Value), và tách riêng các kênh màu để phân tích chi tiết.
* **Áp dụng thuật toán GrabCut:** Thuật toán này giúp tách vùng quả cà chua khỏi nền ảnh, đảm bảo các phân tích tiếp theo chỉ tập trung vào đối tượng cần thiết.
* **Tính toán histogram:** OpenCV hỗ trợ xây dựng các biểu đồ màu cho từng kênh (Hue, Saturation, Value), giúp xác định màu chủ đạo của quả cà chua và đánh giá trạng thái độ chín.
* **Phân tích các vùng bất thường:** Bằng cách so sánh giá trị Hue với các ngưỡng màu sắc chuẩn, OpenCV giúp phát hiện các vùng màu sắc bất thường, hỗ trợ nhận biết quả cà chua có dấu hiệu bị hỏng.
  1. **NumPy**

**NumPy** là một thư viện quan trọng trong lĩnh vực khoa học dữ liệu và tính toán số học. Thư viện này cung cấp các công cụ làm việc với mảng (array) và ma trận, hỗ trợ thực hiện các phép tính phức tạp với hiệu năng cao.

Trong dự án, NumPy được sử dụng để:

* **Xử lý ma trận ảnh:** Dữ liệu hình ảnh trong OpenCV được lưu trữ dưới dạng mảng, và NumPy cung cấp các phép toán nhanh chóng để thao tác trên dữ liệu này.
* **Phân tích pixel:** NumPy hỗ trợ tính toán các giá trị trung bình, tỷ lệ phần trăm, hoặc tìm kiếm các giá trị lớn nhất/nhỏ nhất trong dữ liệu màu sắc.
* **Lọc và phân loại dữ liệu:** Thư viện này giúp lọc bỏ các giá trị không cần thiết (như pixel nền), tập trung vào phân tích vùng ảnh chính chứa quả cà chua.
  1. **Matplotlib**

**Matplotlib** là một thư viện đồ họa phổ biến, được sử dụng để trực quan hóa dữ liệu dưới dạng biểu đồ và đồ thị.

Trong dự án, Matplotlib đóng vai trò quan trọng trong việc:

* **Vẽ biểu đồ histogram:** Trực quan hóa các kênh màu Hue, Saturation và Value dưới dạng biểu đồ, giúp người dùng dễ dàng quan sát sự phân bố màu sắc trong ảnh.
* **Hiển thị ảnh đã qua xử lý:** Các kết quả từ thuật toán GrabCut hoặc ảnh gốc được hiển thị để người dùng có thể so sánh trực quan.
* **Tổng hợp kết quả phân tích:** Kết hợp với giao diện PyQt5, Matplotlib hiển thị các biểu đồ và hình ảnh trong cửa sổ phân tích, giúp người dùng dễ dàng theo dõi.
  1. **Các module tùy chỉnh**
* **Ui\_MainWindow:** Đây là module tự động sinh từ tệp .ui, chịu trách nhiệm định nghĩa và quản lý giao diện chính của ứng dụng. Giao diện này chứa các thành phần chính như vùng chọn ảnh, nút phân tích, và vùng hiển thị kết quả.
  1. **Công cụ phát triển**
* **PyCharm:** IDE PyCharm được sử dụng để phát triển, kiểm tra, và gỡ lỗi mã nguồn. Với tính năng tự động hoàn thành mã, kiểm tra cú pháp, và quản lý thư viện, PyCharm hỗ trợ tối đa cho lập trình viên trong suốt quá trình xây dựng hệ thống.
* **Qt Designer:** Công cụ thiết kế giao diện trực quan, giúp nhanh chóng tạo ra bố cục các cửa sổ và widget cần thiết. Các tệp .ui từ Qt Designer được tích hợp vào dự án thông qua PyQt5.

Sự kết hợp giữa các thư viện, công cụ và module như PyQt5, OpenCV, NumPy, và Matplotlib mang lại sự linh hoạt và hiệu quả cao cho dự án. Những thành phần này không chỉ hỗ trợ tốt trong xử lý ảnh và phân tích màu sắc, mà còn giúp tạo ra một giao diện người dùng trực quan và dễ sử dụng, đáp ứng tốt yêu cầu của bài toán.

**Chương II: Xây dựng chương trình**

1. **Mô tả bài toán**

Trong nông nghiệp hiện đại, việc đánh giá độ chín và tình trạng của trái cây, đặc biệt là cà chua, đóng vai trò quan trọng trong việc quản lý thu hoạch, bảo quản, và phân phối. Tuy nhiên, phương pháp truyền thống dựa vào đánh giá bằng mắt thường không chỉ mất thời gian mà còn có độ chính xác thấp, dễ phụ thuộc vào cảm quan của từng người.

Dự án này đặt mục tiêu xây dựng một hệ thống nhận diện màu sắc tự động để phân loại và đánh giá độ chín của cà chua dựa trên hình ảnh đầu vào. Thông qua việc ứng dụng các thuật toán xử lý ảnh và phân tích dữ liệu, hệ thống không chỉ xác định trạng thái chín mà còn có thể phát hiện các dấu hiệu bất thường, chẳng hạn như hư hỏng hoặc bị nấm mốc.

**1.1. Mục tiêu cụ thể**

1. **Nhận diện vùng chứa quả cà chua**

Sử dụng thuật toán GrabCut để phân vùng và tách đối tượng cà chua ra khỏi nền ảnh, đảm bảo phân tích chính xác.

1. **Phân tích màu sắc:**
   * Áp dụng không gian màu HSV để xác định giá trị Hue (màu sắc chủ đạo), Saturation (độ bão hòa), và Value (độ sáng).
   * Tính toán histogram để phân bố các giá trị màu sắc trong vùng ảnh được tách.
2. **Phân loại độ chín của cà chua:**
   * Dựa vào các ngưỡng giá trị Hue, xác định trạng thái của quả cà chua, ví dụ:
     + **Chín:** Màu đỏ chiếm ưu thế.
     + **Hơi xanh:** Pha trộn giữa màu cam và xanh lá.
     + **Xanh:** Màu xanh lá chiếm đa số.
     + **Bất thường:** Xuất hiện các giá trị màu không hợp lý hoặc mức độ phân bố màu bất đối xứng.
3. **Phát hiện hư hỏng:**

Phân tích tỷ lệ các pixel bất thường trong vùng ảnh (ví dụ: các giá trị Hue không nằm trong khoảng màu tự nhiên của cà chua), từ đó đưa ra cảnh báo nếu quả cà chua có dấu hiệu bị hỏng.

1. **Hiển thị kết quả:**
   * Giao diện trực quan cho phép người dùng tải ảnh, xem kết quả phân loại, và biểu đồ histogram màu sắc.
   * Đề xuất hành động, chẳng hạn loại bỏ cà chua hỏng hoặc lưu trữ dữ liệu để phân tích sau.

**1.2. Đầu vào và đầu ra của bài toán**

* **Đầu vào:**
  + Một hình ảnh kỹ thuật số (thường ở định dạng JPG hoặc PNG) chứa một hoặc nhiều quả cà chua. Hình ảnh có thể có nền phức tạp, chẳng hạn như lá, cành, hoặc bàn tay con người.
  + Kích thước ảnh và chất lượng có thể thay đổi, nhưng cần đảm bảo độ phân giải đủ cao để nhận diện vùng màu chính xác.
* **Đầu ra:**
  + **Thông tin phân tích:**
    - Màu sắc chủ đạo (Hue) của quả cà chua.
    - Trạng thái độ chín, ví dụ: Chín, Hơi xanh, Xanh hoặc Bất thường.
    - Tỷ lệ phần trăm vùng ảnh chứa các giá trị màu bất thường (phát hiện hỏng).
  + **Biểu đồ trực quan:**
    - Biểu đồ histogram của các kênh màu Hue, Saturation, và Value.
    - Ảnh vùng cà chua sau khi được tách nền bằng GrabCut.
  + **Đề xuất hành động:** Cảnh báo nếu quả cà chua bị hỏng và khuyến nghị xử lý.

**1.3. Phạm vi ứng dụng thực tế**

Hệ thống có thể được ứng dụng rộng rãi trong nhiều lĩnh vực, bao gồm:

* **Nông nghiệp:** Giám sát và phân loại cà chua tại các cơ sở thu hoạch, đảm bảo độ đồng đều về chất lượng sản phẩm.
* **Chế biến thực phẩm:** Phân loại nguyên liệu đầu vào để tối ưu hóa quy trình chế biến và bảo quản.
* **Khoa học nghiên cứu:** Thu thập dữ liệu và phân tích đặc điểm màu sắc trong quá trình phát triển của trái cây.

**1.4. Những thách thức và hạn chế**

* **Đa dạng trong hình ảnh đầu vào:** Các hình ảnh có nền phức tạp hoặc điều kiện ánh sáng không đồng đều có thể ảnh hưởng đến độ chính xác của phân tích.
* **Sai lệch trong không gian màu:** Sự khác biệt giữa các thiết bị chụp ảnh (camera, điện thoại) có thể dẫn đến sai lệch trong giá trị màu.
* **Phân loại trạng thái hư hỏng:** Một số quả cà chua có thể có vùng bị hỏng nhỏ, khó phát hiện bằng cách phân tích toàn cục.
* **Hiệu năng xử lý:** Đối với các hình ảnh lớn hoặc có độ phân giải cao, việc áp dụng các thuật toán xử lý ảnh có thể mất thời gian đáng kể.

**1.5. Kỳ vọng của hệ thống**

Hệ thống không chỉ dừng lại ở việc nhận diện và phân loại cà chua mà còn hướng đến xây dựng một giải pháp tự động hóa linh hoạt và dễ sử dụng, giúp người dùng cuối (nông dân, nhà nghiên cứu, doanh nghiệp thực phẩm) tiết kiệm thời gian và nâng cao hiệu quả công việc.

1. **Mô tả thuật toán/mô hình được áp dụng**

Hệ thống sử dụng các thuật toán và mô hình xử lý ảnh cơ bản nhưng hiệu quả để đạt được mục tiêu phân loại độ chín và phát hiện tình trạng bất thường của cà chua. Cụ thể, các bước xử lý bao gồm tách đối tượng khỏi nền bằng **thuật toán GrabCut**, chuyển đổi không gian màu để phân tích màu sắc trong hệ màu **HSV**, và sử dụng histogram để đánh giá sự phân bố màu sắc.

**2.1. Thuật toán GrabCut**

**Thuật toán GrabCut**, được giới thiệu bởi Carsten Rother, Vladimir Kolmogorov và Andrew Blake vào năm 2004, là một thuật toán phân vùng ảnh bán tự động, hiệu quả trong việc tách đối tượng chính ra khỏi nền.

**Nguyên lý hoạt động**

GrabCut sử dụng mô hình đồ thị (graph-based segmentation) kết hợp với Gaussian Mixture Model (GMM) để phân loại các pixel trong ảnh thành hai nhóm:

* **Foreground (FG):** Các pixel thuộc đối tượng cần tách.
* **Background (BG):** Các pixel thuộc nền.

Quy trình cơ bản của GrabCut:

1. **Khởi tạo:**

Người dùng xác định một khung hình chữ nhật bao quanh đối tượng (bounding box). Pixel bên trong khung được đánh dấu là “không xác định”, trong khi các pixel bên ngoài được đánh dấu là “nền”.

1. **Mô hình hóa FG và BG:**

Thuật toán xây dựng hai mô hình GMM dựa trên phân phối màu sắc của các pixel thuộc FG và BG.

1. **Tối ưu hóa:**

Sử dụng thuật toán cắt đồ thị (Graph Cut), GrabCut tối ưu hóa sự phân vùng giữa FG và BG bằng cách cân bằng giữa độ tương đồng màu sắc và tính liên tục của các pixel.

1. **Lặp lại:**

Thuật toán tự động cải thiện kết quả phân vùng qua từng vòng lặp bằng cách cập nhật GMM và đánh giá lại.

**Ưu điểm của GrabCut**

* Hiệu quả và chính xác trong việc tách đối tượng khỏi nền, ngay cả khi nền phức tạp.
* Hoạt động tốt với các đối tượng có biên dạng không đều.
* Có khả năng lặp lại để cải thiện kết quả nếu khung ban đầu không chính xác.

Trong bài toán này, GrabCut được sử dụng để tách riêng quả cà chua ra khỏi nền, đảm bảo các bước phân tích sau đó chỉ tập trung vào đối tượng chính.

**2.2. Chuyển đổi không gian màu HSV**

Sau khi tách được quả cà chua bằng GrabCut, hệ thống chuyển đổi ảnh từ không gian màu BGR (mặc định trong OpenCV) sang không gian màu **HSV**.

**Lý do chọn HSV:**

HSV là một không gian màu trực quan và phù hợp cho các bài toán liên quan đến phân tích màu sắc nhờ tách biệt rõ ràng giữa:

* **Hue (H):** Thông tin về màu sắc chủ đạo (đỏ, xanh lá, xanh dương, v.v.).
* **Saturation (S):** Độ bão hòa của màu (mức độ tinh khiết của màu sắc).
* **Value (V):** Độ sáng của màu.

**Phân tích trong HSV:**

* **Hue:**
  + Xác định màu sắc chủ đạo, từ đó phân loại trạng thái độ chín của cà chua.
  + Ví dụ: Hue từ 0-8 hoặc 330-360 tương ứng với màu đỏ, đặc trưng của cà chua chín.
* **Saturation:**
  + Đo độ tinh khiết của màu sắc, giúp phân biệt quả tươi hoặc đang mất màu (dấu hiệu hư hỏng).
* **Value:**
  + Đánh giá độ sáng tổng thể của vùng ảnh, hỗ trợ phát hiện vùng tối bất thường.

**2.3. Histogram và phân tích màu sắc**

Histogram là công cụ biểu diễn trực quan sự phân bố của các giá trị màu sắc trong vùng ảnh. Trong hệ thống, histogram được tính cho từng kênh H, S, V để đánh giá sự cân đối màu sắc trong quả cà chua.

**Tính toán histogram:**

* Sử dụng cv2.calcHist() để tính toán số lượng pixel cho mỗi giá trị trong từng kênh.
* Loại bỏ các giá trị màu không liên quan (như pixel nền bị nhiễu).

**Ứng dụng của histogram:**

1. **Xác định màu chủ đạo:**
   * Giá trị Hue chiếm ưu thế trong histogram biểu thị màu sắc chính của cà chua.
2. **Đánh giá mức độ chín:**
   * Dựa vào các ngưỡng Hue để phân loại:
     + **0-8 hoặc 330-360:** Cà chua chín (đỏ).
     + **15-30:** Cà chua hơi xanh (cam).
     + **30-60:** Cà chua chưa chín (xanh lá).

*A screenshot of a computer code

Description automatically generated*

1. **Phát hiện bất thường:**
   * Tỷ lệ pixel nằm ngoài khoảng giá trị màu tự nhiên của cà chua có thể là dấu hiệu của hư hỏng hoặc nấm mốc.

A computer code with black text

Description automatically generated with medium confidence

**2.4. Phát hiện hư hỏng**

Hư hỏng được phát hiện bằng cách phân tích histogram để tìm các giá trị bất thường trong kênh Hue. Các vùng màu đen, xám hoặc có độ bão hòa thấp là dấu hiệu của nấm mốc hoặc vùng bị hỏng.

A computer screen shot of text

Description automatically generated

Cụ thể:

* Tính tỷ lệ phần trăm pixel có giá trị Hue bất thường so với tổng số pixel trong vùng cà chua.
* Nếu tỷ lệ bất thường vượt quá một ngưỡng nhất định ( 5%), hệ thống cảnh báo quả cà chua có thể bị hỏng.

**2.5. Quy trình tổng quan**

1. **Nhập ảnh:** Người dùng tải hình ảnh chứa quả cà chua.
2. **Tách nền:** Sử dụng GrabCut để tách riêng vùng quả cà chua.
3. **Chuyển đổi màu:** Chuyển không gian màu sang HSV để phân tích.
4. **Tính toán và phân tích:**
   * Xác định màu chủ đạo bằng histogram Hue.
   * Phân loại trạng thái độ chín dựa trên giá trị Hue.
   * Phát hiện vùng bất thường và đánh giá tình trạng hư hỏng.
5. **Hiển thị kết quả:** Trả về thông tin chi tiết, biểu đồ histogram, và ảnh đã xử lý.

Hệ thống này kết hợp nhiều bước xử lý ảnh đơn giản nhưng hiệu quả, tận dụng tốt các công cụ mạnh mẽ của OpenCV và khả năng trực quan hóa của Matplotlib.

1. **Cấu trúc mã nguồn và các file mã nguồn quan trọng**

Cấu trúc mã nguồn của dự án này được xây dựng một cách có tổ chức, kết hợp giữa logic xử lý và giao diện đồ họa người dùng (GUI) bằng thư viện PyQt5. Điều này đảm bảo tính rõ ràng, dễ bảo trì, và có khả năng mở rộng cho các tính năng trong tương lai. Các thành phần được chia nhỏ, mỗi file mã nguồn phục vụ một vai trò cụ thể.

* 1. **Cấu trúc tổng quan**

**Danh sách các file mã nguồn quan trọng**

1. **main.py**:
   * Là file chính, chịu trách nhiệm khởi chạy ứng dụng và tích hợp giữa giao diện với logic xử lý ảnh.
   * Tích hợp các thư viện cần thiết như OpenCV và NumPy để thực hiện các phép toán liên quan đến xử lý hình ảnh.
   * Cung cấp các chức năng như tải ảnh, phân tích vùng quan tâm (ROI), và hiển thị kết quả trên giao diện.
2. **ui\_main\_window.py**:
   * Được tự động sinh ra từ file thiết kế giao diện main\_window.ui bằng công cụ pyuic5.
   * Định nghĩa cấu trúc và bố cục giao diện đồ họa của ứng dụng, bao gồm các nút, nhãn, và bố trí tổng thể.
   * Không chứa logic xử lý mà chỉ cung cấp các thành phần giao diện để sử dụng trong file main.py.
3. **main\_window.ui** (không trực tiếp xuất hiện trong mã Python, nhưng là nguồn gốc của ui\_main\_window.py):
   * Là file thiết kế giao diện đồ họa, được tạo bằng PyQt5 Designer.
   * Lưu trữ thông tin về giao diện dưới dạng XML và có thể được chỉnh sửa trực quan.
   1. **Chi tiết từng file mã nguồn**

**1. File main.py**

**Vai trò và nhiệm vụ**

File main.py là trung tâm của ứng dụng, đảm nhận các nhiệm vụ sau:

* **Khởi tạo giao diện người dùng**:

Tải giao diện từ lớp Ui\_MainWindow (được định nghĩa trong ui\_main\_window.py) và hiển thị cửa sổ chính của ứng dụng.

* **Quản lý sự kiện giao diện**:

Liên kết các nút bấm với các chức năng tương ứng, chẳng hạn như:

* + Nút "Insert Image" để người dùng chọn ảnh từ máy tính.
  + Nút "Positioning" để vẽ vùng quan tâm (ROI) trên ảnh.
  + Nút "Analyze" để phân tích và hiển thị kết quả.
* **Tích hợp xử lý ảnh**:

Sử dụng thư viện OpenCV để xử lý ảnh, bao gồm:

* + Chuyển đổi không gian màu.
  + Phân đoạn ảnh bằng thuật toán GrabCut.
  + Phân tích histogram màu để xác định độ chín của đối tượng (trong trường hợp này là cà chua).

**Cấu trúc lớp và phương thức**

Lớp chính trong file này là MainWindow, kế thừa từ QMainWindow.

**Phương thức khởi tạo**

def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.ui = Ui\_MainWindow()  
 self.ui.setupUi(self)

* Kết nối giao diện được định nghĩa trong ui\_main\_window.py.
* Thiết lập các sự kiện cho các nút trong giao diện.

**Các phương thức chính**

1. **insert\_image()**
   * **Mô tả**: Cho phép người dùng chọn một tệp ảnh từ máy tính và hiển thị ảnh trên giao diện.
   * **Chi tiết hoạt động**:
     + Mở hộp thoại tập tin (QFileDialog) để người dùng chọn ảnh.
     + Ảnh được tải và hiển thị trên lbl\_img (QLabel hiển thị ảnh).
     + Lưu ảnh vào biến nội bộ để xử lý sau này.

A computer screen shot of text

Description automatically generated

1. **activate\_positioning()**
   * **Mô tả**: Kích hoạt chế độ cho phép người dùng chọn vùng quan tâm (ROI) bằng chuột.
   * **Chi tiết hoạt động**:
     + Theo dõi các sự kiện chuột để xác định và vẽ hình chữ nhật đại diện cho vùng ROI.
     + Sử dụng các sự kiện như mousePressEvent, mouseMoveEvent, và mouseReleaseEvent.
     + Lưu tọa độ ROI vào một biến để sử dụng trong các bước xử lý tiếp theo.
2. **analyze\_image()**
   * **Mô tả**: Phân tích vùng ROI được chọn, áp dụng các thuật toán xử lý ảnh để trích xuất thông tin.
   * **Chi tiết hoạt động**:
     + Chuyển đổi vùng ROI sang không gian màu HSV.
     + Tính toán và hiển thị các biểu đồ màu sắc để phân tích đặc điểm độ chín của đối tượng.
     + Phân đoạn hình ảnh bằng thuật toán GrabCut để chỉ tập trung vào vùng cần thiết.
3. **show\_image(image)**
   * **Mô tả**: Hiển thị hình ảnh (gốc hoặc đã qua xử lý) trên nhãn lbl\_img.
   * **Chi tiết hoạt động**:
     + Chuyển đổi hình ảnh từ định dạng NumPy (BGR) sang định dạng QImage để tương thích với PyQt.
     + Hiển thị trên QLabel.

**2. File ui\_main\_window.py**

**Vai trò và nhiệm vụ**

* File ui\_main\_window.py được tạo tự động từ PyQt5 Designer.
* Định nghĩa giao diện đồ họa của ứng dụng:
  + Các thành phần chính như nút bấm, nhãn hiển thị ảnh, và bố cục giao diện.
  + Không chứa logic xử lý, chỉ chịu trách nhiệm tạo giao diện.

**Cấu trúc giao diện**

1. **Nút bấm (QPushButton)**:
   * **btn\_insert**: Dùng để chọn ảnh đầu vào.
   * **btn\_positioning**: Kích hoạt chế độ vẽ vùng ROI.
   * **btn\_analyze**: Thực hiện phân tích ảnh.
2. **Nhãn hiển thị ảnh (QLabel)**:
   * **lbl\_img**: Hiển thị ảnh được chọn hoặc ảnh đã qua xử lý.
   * Mặc định hiển thị dòng chữ "No Image" nếu chưa có ảnh nào được tải.
3. **Bố cục lưới (QGridLayout)**:
   * Sắp xếp các thành phần giao diện theo dạng lưới để tối ưu không gian hiển thị.

**Liên kết với main.py**

* Các thành phần giao diện như nút bấm, nhãn hiển thị được truy cập từ main.py qua lớp Ui\_MainWindow.
* Không có logic xử lý, chỉ cung cấp các widget để sử dụng.
  1. **Ưu điểm của cấu trúc mã nguồn**

1. **Tách biệt rõ ràng giữa giao diện và logic**:
   * Giao diện được định nghĩa trong ui\_main\_window.py, giúp dễ dàng sửa đổi mà không ảnh hưởng đến logic xử lý trong main.py.
2. **Dễ dàng mở rộng**:
   * Có thể thêm chức năng mới trong main.py mà không làm ảnh hưởng đến cấu trúc giao diện.
3. **Sử dụng công cụ trực quan**:
   * PyQt5 Designer giúp tiết kiệm thời gian thiết kế giao diện, đảm bảo tính thẩm mỹ và nhất quán.
4. **Tái sử dụng mã nguồn**:
   * Các chức năng được chia nhỏ và có thể được sử dụng lại trong các dự án khác.

Cấu trúc mã nguồn được thiết kế để đảm bảo tính dễ hiểu, linh hoạt, và mạnh mẽ trong việc xử lý hình ảnh và hiển thị trên giao diện.

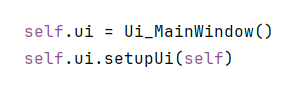
1. **Thiết kế giao diện chương trình**

Giao diện chương trình được xây dựng bằng công cụ **PyQt5 Designer** và lưu trữ trong file main\_window.ui. Đây là một giao diện đồ họa trực quan, dễ sử dụng, được thiết kế nhằm đáp ứng các chức năng chính như chọn ảnh, vẽ vùng quan tâm (ROI), và phân tích hình ảnh. Giao diện được tối ưu để hiển thị ảnh và cung cấp các nút thao tác rõ ràng cho người dùng.

* 1. **Thành phần chính của giao diện**

1. **Cửa sổ chính (QMainWindow)**
   * Là giao diện chính của chương trình, nơi người dùng tương tác với các chức năng.
   * Kích thước mặc định: **800x600** pixel, đảm bảo đủ không gian hiển thị ảnh và các nút chức năng.
2. **Các thành phần giao diện**
   * **Nút bấm (QPushButton)**
     + **Insert Image**: Cho phép người dùng tải ảnh từ máy tính.
       - **Vị trí**: Góc trên bên trái của cửa sổ.
       - **Kích thước**: 111x41 pixel.
       - **Vai trò**: Kích hoạt hộp thoại chọn ảnh, sau đó hiển thị ảnh được chọn.
     + **Positioning**: Kích hoạt chế độ vẽ vùng quan tâm (ROI).
       - **Vị trí**: Góc dưới bên trái (150x470 pixel).
       - **Kích thước**: 161x51 pixel.
       - **Vai trò**: Cho phép người dùng chọn vùng cần phân tích bằng cách vẽ hình chữ nhật trên ảnh.
     + **Analyze**: Phân tích hình ảnh trong vùng đã chọn.
       - **Vị trí**: Góc dưới bên phải (470x470 pixel).
       - **Kích thước**: 161x51 pixel.
       - **Vai trò**: Áp dụng thuật toán xử lý ảnh và hiển thị kết quả phân tích.
   * **Nhãn hiển thị ảnh (QLabel)**
     + **Tên**: lbl\_img.
     + **Vị trí**: Chiếm phần lớn diện tích trung tâm cửa sổ, bên trong QGridLayout.
     + **Vai trò**: Hiển thị hình ảnh gốc, ảnh đã qua xử lý, hoặc thông báo khi chưa có ảnh ("No Image").
     + **Kích thước tự động**: Tùy thuộc vào kích thước ảnh và cửa sổ.
   * **Bố cục lưới (QGridLayout)**
     + Được sử dụng để sắp xếp các thành phần giao diện trong không gian hiển thị chính.
     + **Vai trò**: Đảm bảo ảnh được hiển thị cân đối và chiếm toàn bộ không gian có sẵn.
   * **Thanh trạng thái (QStatusBar)**
     + **Vai trò**: Hiển thị thông báo ngắn gọn về trạng thái hiện tại của chương trình (ví dụ: "Image loaded successfully").
3. **Menu Bar (QMenuBar)**
   * Hiện tại chưa được sử dụng, nhưng có thể mở rộng trong tương lai để thêm các menu chức năng (như "File", "Edit", "Help").
   1. **Cách hiển thị giao diện**

Giao diện được tạo tự động từ file main\_window.ui nhờ công cụ PyQt5 (pyuic5), và mã hóa trong file ui\_main\_window.py. Giao diện này được gọi và hiển thị trong main.py như sau:



* 1. **Đặc điểm nổi bật của thiết kế giao diện**

1. **Giao diện đơn giản, trực quan**:
   * Các nút chức năng được bố trí rõ ràng, dễ dàng truy cập, ngay cả với người dùng không quen thuộc với xử lý hình ảnh.
2. **Không gian hiển thị ảnh lớn**:
   * Ảnh chiếm phần lớn diện tích cửa sổ, đảm bảo người dùng có thể quan sát chi tiết kết quả.
3. **Tính mở rộng cao**:
   * Giao diện được thiết kế linh hoạt, có thể thêm các thành phần mới (như biểu đồ, bảng số liệu) mà không làm ảnh hưởng đến cấu trúc tổng thể.
4. **Phù hợp với quy trình công việc**:
   * Các chức năng "Insert Image", "Positioning", và "Analyze" được sắp xếp theo thứ tự logic, tương ứng với từng bước trong quy trình phân tích hình ảnh.
   1. **Gợi ý cải tiến giao diện**
5. **Thêm thanh công cụ (Toolbar)**
   * Cung cấp các biểu tượng nhanh cho các chức năng chính như mở ảnh, lưu ảnh, hoặc hoàn tác (Undo).
6. **Cửa sổ thông báo kết quả**
   * Hiển thị thông tin chi tiết về kết quả phân tích dưới dạng bảng hoặc biểu đồ.
7. **Hỗ trợ đa ngôn ngữ**
   * Sử dụng công cụ dịch của PyQt5 (QTranslator) để hỗ trợ người dùng với các ngôn ngữ khác nhau.
8. **Cải thiện thẩm mỹ**
   * Tùy chỉnh giao diện với **QSS (Qt Style Sheets)** để tạo cảm giác chuyên nghiệp hơn, như thay đổi màu nền, phông chữ, và hiệu ứng nút bấm.

Thiết kế giao diện chương trình tập trung vào sự tiện lợi và dễ sử dụng, nhưng vẫn đảm bảo khả năng mở rộng trong tương lai. Cách bố trí và các thành phần được chọn lọc để phù hợp với ứng dụng xử lý hình ảnh, mang lại trải nghiệm tối ưu cho người dùng.

**Chương III: Kết quả thực nghiệm**

1. **Kiểm thử các chức năng**
2. **Kết quả thực nghiệm**

**Chương IV: Kết luận**

1. **Đánh giá tổng quan**
2. **Dự kiến phát triển trong tương lai**

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated