1. Khái quát về xử lí ảnh và ứng dụng xử lí ảnh để phân loại sản phẩm.
   1. Khái quát về xử lí ảnh.

* Khái niệm

Xử lý ảnh là một lĩnh vực nghiên cứu và ứng dụng của khoa học máy tính và kỹ thuật số, tập trung vào việc phân tích và thao tác hình ảnh kỹ thuật số để trích xuất thông tin hữu ích hoặc cải thiện chất lượng hình ảnh.

Cụ thể, xử lý ảnh liên quan đến một loạt các thao tác và kỹ thuật để biến đổi ảnh nhằm:

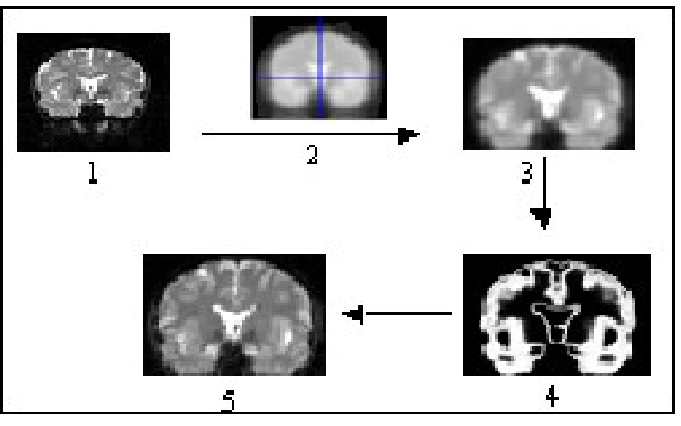
* **Cải thiện chất lượng ảnh**: Loại bỏ nhiễu, tăng cường độ tương phản, làm sắc nét hoặc làm mịn ảnh.
* **Trích xuất thông tin**: Phân tích các đối tượng trong ảnh, nhận diện hình dạng, phát hiện biên và các đặc trưng quan trọng.
* **Nén ảnh**: Giảm kích thước tệp ảnh mà không làm mất đi quá nhiều thông tin, thuận tiện cho lưu trữ và truyền tải.

Các bước chính trong xử lý ảnh thường bao gồm:

* **Thu thập và tiền xử lý dữ liệu ảnh**: Thu thập ảnh từ các nguồn khác nhau (máy ảnh, máy quét, cảm biến) và thực hiện các bước tiền xử lý như chuyển đổi định dạng, điều chỉnh độ sáng, độ tương phản, loại bỏ nhiễu.
* **Trích xuất đặc trưng**: Xác định và trích xuất các đặc trưng quan trọng từ ảnh như các điểm đặc trưng, cạnh, biên, hình dạng, màu sắc.
* **Phân tích và nhận diện**: Sử dụng các kỹ thuật học máy và học sâu để phân tích và nhận diện các đối tượng trong ảnh.
* Ý nghĩa.

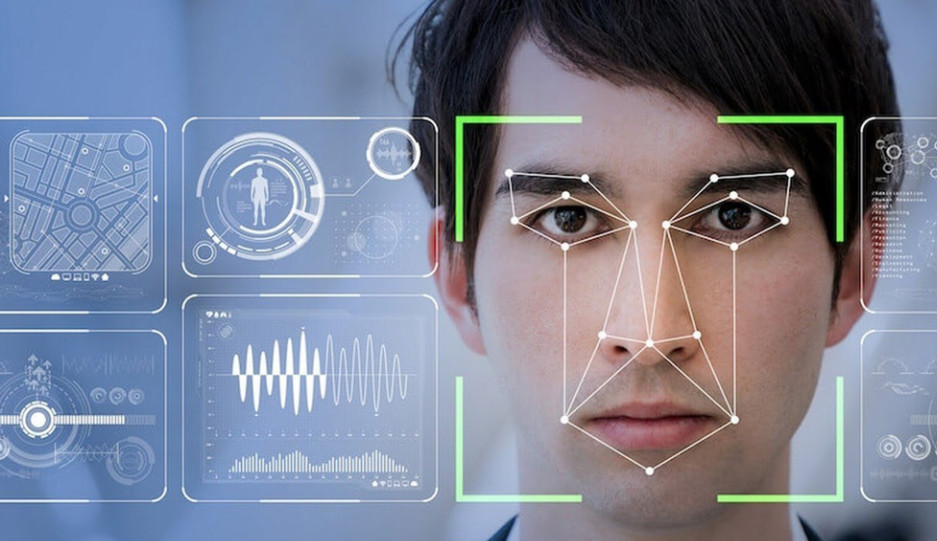
Xử lý ảnh và nhận dạng hình dạng vật thể ngày càng trở nên quan trọng trong nhiều lĩnh vực của cuộc sống hiện đại. Dưới đây là một số nhu cầu thực tế quan trọng:

* **Y tế**: Trong lĩnh vực y tế, việc xử lý ảnh y khoa giúp các bác sĩ chẩn đoán bệnh một cách chính xác hơn. Ví dụ, công nghệ xử lý ảnh giúp phát hiện sớm các bệnh lý như ung thư, thông qua phân tích hình ảnh từ máy MRI, CT hay X-quang. Một nghiên cứu gần đây cho thấy, sử dụng AI trong phân tích ảnh y khoa có thể cải thiện độ chính xác chẩn đoán lên đến 95% so với 85% khi chỉ có bác sĩ thực hiện .



Hình 1.1.1. Quy trình phân đoạn ảnh MR.[[1]](#footnote-1)[1]

* **An ninh**: Hệ thống camera giám sát và nhận diện khuôn mặt ngày càng được triển khai rộng rãi để nâng cao an ninh công cộng. Các công nghệ này giúp nhận dạng và theo dõi các đối tượng khả nghi trong đám đông, giảm thiểu tội phạm và quản lý an ninh hiệu quả hơn. Một ví dụ điển hình là việc triển khai hệ thống nhận diện khuôn mặt tại các sân bay lớn trên thế giới, giúp tăng tốc độ xử lý hành khách và đảm bảo an ninh.



Hình 1.1.2. Hệ thống nhận diện khuôn mặt.[[2]](#footnote-2)[2]

* **Sản xuất công nghiệp**: Trong các nhà máy sản xuất, việc sử dụng hệ thống kiểm tra chất lượng tự động dựa trên xử lý ảnh giúp phát hiện các sản phẩm lỗi một cách nhanh chóng và chính xác. Điều này không chỉ nâng cao chất lượng sản phẩm mà còn giảm thiểu chi phí sản xuất do loại bỏ được các sản phẩm lỗi ngay từ sớm. Các hệ thống này có thể phát hiện các khuyết điểm nhỏ trên dây chuyền sản xuất với độ chính xác lên đến 98% .

A bar code with green and red lines

Description automatically generated

Hình 1.1.3. Barcode và QRcode trên sản phẩm. [[3]](#footnote-3)[3]

* **Giao thông thông minh**: Xử lý ảnh được ứng dụng trong các hệ thống giao thông thông minh để giám sát và quản lý luồng giao thông. Các hệ thống này có thể phát hiện vi phạm giao thông, theo dõi lưu lượng xe cộ, và điều chỉnh đèn giao thông một cách tự động để giảm tắc nghẽn. Tại các thành phố lớn như Singapore, hệ thống giao thông thông minh đã giúp giảm thời gian chờ đợi của người dân tại các ngã tư lên đến 20% .



Hình 1.1.4. Nhận diện biển số xe.[[4]](#footnote-4)[4]

* Các phương pháp xử lí ảnh và kỹ thuật nhận dạng phổ biến.

Trong xử lý ảnh, có nhiều phương pháp khác nhau được sử dụng để đạt được các mục tiêu cụ thể:

* **Làm mịn ảnh**: Giảm nhiễu và làm mịn các chi tiết không mong muốn trong ảnh. Ví dụ: sử dụng bộ lọc Gaussian hoặc Median.
* **Phát hiện biên**: Xác định các cạnh và biên của các vật thể trong ảnh. Các kỹ thuật phổ biến gồm Sobel, Canny, và Laplacian.
* **Phân vùng ảnh**: Chia ảnh thành các phần hoặc vùng có đặc điểm tương đồng. Kỹ thuật K-means và Watershed là hai ví dụ điển hình.

Nhận dạng hình dạng là một phần quan trọng của xử lý ảnh, nhằm xác định và phân loại các hình dạng của vật thể trong ảnh. Các kỹ thuật phổ biến bao gồm:

* **Sử dụng đặc trưng**: Trích xuất các đặc trưng quan trọng từ ảnh như các điểm đặc trưng, góc, cạnh, và sử dụng chúng để nhận dạng hình dạng.
* **Phân loại dựa trên học máy**: Áp dụng các thuật toán học máy như SVM, KNN, hoặc Decision Trees để phân loại hình dạng dựa trên đặc trưng trích xuất.
* **Học sâu (Deep Learning)**: Sử dụng các mạng nơ-ron tích chập (CNN) để tự động trích xuất đặc trưng và nhận dạng hình dạng với độ chính xác cao.
  1. Ứng dụng xử lí ảnh để phân loại sản phẩm trong sản xuất**.**
     1. Tham khảo sản phẩm trên thị trường.

Việc phân loại sản phẩm chính xác và hiệu quả trong công nghiệp là một nhu cầu hết sức thiết yếu. Để đáp ứng nhu cầu khó khăn này, công nghệ thị giác máy đã nổi lên như một công cụ chủ chốt không thể thiếu trong việc kiểm tra và phân loại sản phẩm. Thị giác máy không chỉ giúp tiết kiệm thời gian và chi phí sản xuất mà còn cho ra các thông tin chi tiết về sản phẩm.

Ví dụ: 1 thiết bị xử lí ảnh trên thị trường.



Hình 1.2.1. Thiết bị xử lí ảnh công nghiệp - CVS.[[5]](#footnote-5)[5]

Máy xử lý ảnh công nghiệp (CVS) là dòng máy được tích hợp sẵn phần mềm chuyên dụng cho việc xử lý ảnh công nghiệp. Máy có thể kết nối với 1 hoặc 2 camera cùng lúc và thực hiện các nhiệm vụ: phát hiện thiếu sản phẩm, phát hiện các lỗi ngoại dạng trên bề mặt sản phẩm hay đọc barcode .v.v...

Máy hỗ trợ giao tiếp Modbus RTU, TCP, RS-232 và các kênh giao tiếp I/O giúp thu thập tín hiệu cảm biến và xuất tín hiệu phân loại sản phẩm.

Vỏ máy là nhôm nguyên khối với khe tản nhiệt được tối ưu giúp máy luôn mát và hoạt động ổn định.

* Thông số kỹ thuật.
* Bộ xử lý: 1.92 GHz, Quad Core Intel Cherry Trail
* RAM: 2Gb/4Gb
* SSD: 32Gb/64Gb
* Số cổng kết nối Camera: 01 LAN, 01 USB3.0, 02 USB2.0
* Cổng truyền thông: 01 Rx-Tx
* Phần mềm hỗ trợ truyền thông Modbus RTU, TCP, RS232
* Có thể thay đổi Model kiểm tra: có
* Có thể nâng cấp phần mềm: có
* Phần mềm tích hợp:
* Phần mềm kiểm tra ngoại dạng sản phẩm (MVS20)
* Phần mềm đọc barcode và kiểm tra lỗi barcode
* Phần mềm đọc barcode và phân loại sản phẩm
* Phần mềm kiểm tra nét mực in (thiếu mực, đứt nét, nhòe mực)/ Customize theo yêu cầu khách hàng
* Phần mềm kiểm tra màu sắc và phân loại sản phẩm
* Kích thước: 45x85x145
* Khối lượng: 800gram
* Chất liệu vỏ: nhôm nguyên khối.
  + 1. Xây dựng chương trình xử lí ảnh.
* Tính thực tế.
* *Xác định mục tiêu cụ thể, tối ưu hóa cho hệ thống:* Xây dựng thuật toán xử lí ảnh xung quanh 1 vấn đề cụ thể như sản xuất, y tế, nông nghiệp hay an ninh. Thiết kế 1 hệ thống riêng cho từng vấn đề này nhằm đảm bảo việc tối ưu hóa hiệu suất và tốc độ xử lí.
* *Hệ thống hoạt động độc lập:* Hệ thống hoạt động độc lập, có thể triển khai trên nhiều nền tảng, và thiết bị khác nhau. Có thể kết nối tương thích với các thiết bị và nền tảng khác.
* *Bảo mật cho hệ thống:* Hệ thống được phát triển hoàn toàn dựa trên lý thuyết và không sử dụng những thư viện có sẵn hiện nay. Đảm bảo về mặt bản quyền và tính bảo mật của hệ thống.

Giải quyết các vấn đề nghiên cứu trên không chỉ giúp phát triển một ứng dụng xử lý ảnh xám hiệu quả và chất lượng mà còn đảm bảo rằng ứng dụng của bạn có thể hoạt động ổn định, bảo mật và dễ dàng tích hợp vào các hệ thống khác.

* Thông tin ảnh đầu vào:
* Độ phân giải ảnh: 640 x 480 (px).
* Màu sắc: ảnh xám (Giá trị độ sáng từ 0 đến 255).
* Vật thể: các loại hộp carton đã được gắn mác.
* Mục tiêu của hệ thống.
* Xác định được hình dạng đơn giản (hình chữ nhật, vuông, tròn, tam giác), kích thước, trọng tâm và góc nghiên của vật từ đó có thể ứng dụng cho việc cầm nắm vật chính xác và phân loại sản phẩm.
* Xử lí ảnh nhanh, chính xác bằng cách sử dụng các phương pháp tiên tiến và tối ưu để đảm bảo xử lí ảnh nhanh chóng và đáng tin cậy. Các phương pháp này sẽ được thiết kế để áp dụng các phép biến đổi, phân tích và nhận dạng đối tượng trên ảnh một cách chính xác và hiệu quả.
* Đơn giản hóa quá trình xử lí ảnh, tạo ra một hệ thống tự động hóa, không cần phải nhập thông số cụ thể từ người dùng. Điều này đòi hỏi phải sử dụng thuật toán thông minh, có khả năng xác định từ dữ liệu và tự động tìm ra các đặc trưng quan trọng trong ảnh. Hệ thống sẽ thực hiện các phép biến đổi và phân tích ảnh một cách tự động và đáng tin cậy.

1. Cơ sở lý thuyết.

Từ yêu cầu muốn xác định được hình dạng, kích thước, trọng tâm và góc nghiêng của vật, cần có các phương pháp xử lí bao gồm:

* Phương pháp trích xuất các đặc trưng của vật.
* Phương pháp tách biên vật thể.
* Phương pháp lọc nhiễu.
* Phương pháp phân đoạn ảnh.
  1. Phương pháp trích xuất các đặt trưng.
* Thuật toán Hough Transform.

1. Định nghĩa.

Hough Transform (HT) là một kỹ thuật phát hiện các hình dạng cơ bản trong hình ảnh, chẳng hạn như đường thẳng, đường tròn, và đường ellipse, bằng cách biến đổi không gian điểm ảnh thành không gian tham số. Kỹ thuật này đặc biệt hiệu quả trong việc phát hiện các đường thẳng và đường tròn trong ảnh nhiễu.

Sau khi có thông tin về đường thẳng, tròn, elip có thể trích xuất các đặt trưng từ ảnh như hình học, kích thước, trọng tâm và góc nghiên của vật, cần phải xác định được số cạnh, góc nghiêng hợp bởi cạnh và phương ngang, số đỉnh và tọa độ của mỗi đỉnh trong hình.

1. Tính chất.

* *Phát Hiện Hình Dạng Hình Học Cụ Thể.*
* Đường Thẳng: Hough Transform có thể phát hiện các đường thẳng bằng cách chuyển đổi từ không gian tọa độ Cartesian sang không gian tham số (𝜌,𝜃).
* Đường Tròn: Có thể phát hiện các hình tròn bằng cách sử dụng không gian tham số (𝑎,𝑏,𝑟), trong đó (a,b) là tọa độ tâm và 𝑟 là bán kính.
* Hình Dạng Khác: Có thể mở rộng để phát hiện các hình dạng khác như elip, parabol, và các đường cong tham số khác.
* *Tính Bất Biến Đối Với Nhiễu.*
* Ít Nhạy Cảm Với Nhiễu: Hough Transform tích lũy bằng chứng về sự hiện diện của các hình dạng cụ thể trong không gian tham số, làm cho nó ít nhạy cảm với nhiễu cục bộ trong hình ảnh.
* *Độc Lập Với Tỷ Lệ và Xoay.*
* Bất Biến Với Tỷ Lệ và Xoay: Có khả năng phát hiện các hình dạng mục tiêu bất kể kích thước và hướng của chúng trong hình ảnh. Điều này làm cho Hough Transform trở nên linh hoạt trong nhiều ứng dụng khác nhau.
* *Tốn Tài Nguyên Tính Toán.*
* Yêu Cầu Cao Về Tài Nguyên: Đòi hỏi bộ nhớ và thời gian tính toán lớn, đặc biệt khi tìm kiếm các hình dạng phức tạp trong không gian tham số cao. Điều này có thể hạn chế hiệu quả của nó trong các ứng dụng yêu cầu xử lý thời gian thực hoặc trên các thiết bị tài nguyên hạn chế.
* *Không Phù Hợp Cho Các Hình Dạng Tự Do.*
* Hạn Chế Với Hình Dạng Tự Do: Chủ yếu hiệu quả với các hình dạng hình học có tham số rõ ràng. Không phù hợp cho việc phát hiện các hình dạng tự do hoặc không có tham số xác định.
* *Khả Năng Mở Rộng.*
* Linh Hoạt và Mở Rộng: Có thể được điều chỉnh và mở rộng để phát hiện các hình dạng không tiêu chuẩn bằng cách thay đổi không gian tham số và các phương trình hình học tương ứng.

1. Giải thích thuật toán.

Yêu cầu ảnh đầu vào của thuật toán Hough Transform là ảnh nhị phân thể hiện biên dạng của vật thể, từ đó dựa trên những phép biến đổi để xác định những hình ảnh cạnh đó là đường thẳng hoặc cong.

Trường hợp phổ biến được sử dụng của phép biến đổi Hough Transform là phát hiện ra những đường thẳng. Đường thẳng **y = ax + b** có thể được biểu diễn dưới dạng một điểm (a,b) trong không gian tham số. Tuy nhiên, có một vấn đề là nếu ở dạng này chúng sẽ làm phát sinh ra các giá trị không giới hạn của tham số độ dốc **a**. Do đó, vì lý do tính toán phải sử dụng phương trình đường thẳng dưới dạng hệ tọa độ cực.

r = x. cos(θ) + y.sin(θ) (2.1)



Hình 1. Đường thẳng được biểu diễn thông qua r và θ.

Với phương trình (1), cho 1 cặp số (x,y) mà tại đó độ sáng I(x,y) = 255 với θ phân bố từ 0 đến 2π, sẽ vẽ được 1 đường hình sin. Ví dụ, đối với x0 = 8 và y0 = 6, khi vẽ trên mặt phẳng (r - θ) nhận được:

A graph with a curve

Description automatically generated

Hình 2.

Thực hiện thao tác trên cho tất cả các điểm ảnh có độ sáng I = 255 trong cùng 1 tấm ảnh. Nếu đường cong của hai điểm đó giao nhau trong mặt phẳng (r - θ), điều đó có nghĩa là cả 2 điểm đều thuộc về cùng 1 đường thẳng . Ví dụ, làm theo ví dụ ở trên và vẽ cho 2 điểm nữa: x1 = 4, y1 = 9 và x2 = 4, y2 = 9, nhận được kết quả sau:

A graph with colored lines

Description automatically generated

Hình 3.

3 đường giao nhau tại (0.925 , 9.6), tọa độ này đại diện cho các tham số (r , θ). Từ cặp tham số (r , θ) có thể đưa vào phương trình (1) để trở thành 1 phương trình đường thẳng đi qua 3 điểm (x0 , y0), (x1 , y1), (x2 , y2).

Tóm lại, một đường có thể được phát hiện bằng cách tìm số giao điểm giữa các đường hình sin ở trên. Càng nhiều đường sin giao nhau tại 1 điểm có nghĩa là đường thẳng đó càng đi qua nhiều điểm. Từ đó có thể sử dụng một ngưỡng xác định cần thiết để phát hiện 1 đường.

Những gì thuật toán Hough Transform làm là nó theo giao điểm giữa các đường sin của mọi điểm ảnh . Nếu số lượng đường giao tại 1 điểm cao hơn *số ngưỡng ,* thì khai báo nó là một đường thẳng với cặp tham số (r , θ) của điểm giao.

* 1. Phương pháp nhận dạng đường biên.
* Thuật toán nhận dạng đường biên canny.

1. Định nghĩa.

Thuật toán Canny là một trong những phương pháp phát hiện biên phổ biến nhất trong xử lý ảnh, được phát triển bởi John F. Canny vào năm 1986. Mục tiêu của thuật toán Canny là xác định các biên rõ ràng trong ảnh bằng cách giảm thiểu ảnh hưởng của nhiễu trong khi tối đa hóa khả năng phát hiện biên.

1. Tính chất.

* *Độ Chính Xác Tốt.*
* Thuật toán Canny được thiết kế để tối đa hóa khả năng phát hiện các biên thực sự trong ảnh. Nó đảm bảo rằng tất cả các biên quan trọng đều được phát hiện.
* *Định Vị Tốt .*
* Thuật toán Canny xác định vị trí của các biên với độ chính xác cao. Vị trí của các biên được tính toán một cách chính xác mà không bị dịch chuyển bởi nhiễu.
* *Đáp Ứng Đơn Lập.*
* Thuật toán Canny đảm bảo rằng mỗi biên chỉ được phát hiện một lần. Điều này giúp loại bỏ các biên giả và giảm thiểu nhiễu.
* *Khả Năng Chống Nhiễu.*
* Bộ lọc Gaussian trong bước đầu tiên của thuật toán giúp giảm thiểu nhiễu trong ảnh, do đó cải thiện chất lượng của biên được phát hiện.

1. Giải thích thuật toán.

Thuật toán phát hiện cạnh Canny bao gồm 5 bước:

* Giảm nhiễu.
* Tính toán Gradient độ xám của ảnh.
* Áp dụng Non-maximum suppression.
* Ngưỡng kép.
* Theo dõi cạnh bằng độ trễ.

Một điều quang trọng cần lưu ý khi thực hiện thuật toán phát hiện cạnh Canny là bức ảnh đầu vào phải ở thang độ xám trước khi thực hiện.

* Bước 1: Giảm nhiễu.

Có nhiều cách giúp làm giảm nhiễu, cách phổ biến thường gặp là sử dụng bộ lọc Gaussian giúp làm mịn ảnh. Để thực hiện bộ lọc Gaussian, kỹ thuật tích chập hình ảnh được áp dụng với Gaussian Kernel. Kích thước kernel phụ thuộc vào hiệu ứng làm mờ mong muốn. Về cơ bản, kernel càng nhỏ hiệu quả làm mờ càng ít.

Phương trình cho một kernel bộ lọc Gaussian có kích thước (2k +1, 2k+1):

Hij = exp(−) ; 1< i,j<(2k+1)

A comparison of a person

Description automatically generated

Hình 4. Ảnh gốc (trái) – Hình ảnh bị làm mờ với bộ lọc Gaussian (sigma = 1.4 và kích thước kernel là 5×5)[[6]](#footnote-6)[6]

* Bước 2: Tính toán Gradient độ xám.
* Sau khi làm mờ, tính Gradient của ảnh để phát hiện các thay đổi đột ngột trong cường độ sáng, từ đó xác định được các biên.
* Sử dụng các bộ lọc Sobel để tính Gradient theo hướng x và y:

Gx = . I(x,y) (2.2)

Gy = . I(x,y) (2.3)

Với: Gx: Gradient theo hướng x.

Gy: Gradient theo hướng y.

I(x,y): giá trị độ sáng tại điểm có tọa độ (x,y)

* Tính độ lớn và hướng Gradient:

G = (2.4)

𝜃 = arctan () (2.5)

A comparison of a person

Description automatically generated

Hình 5. Hình ảnh bị mờ (trái) – Cường độ gradient (phải)[[7]](#footnote-7)[7]

* Bước 3: Áp dụng Non-maximum suppression.
* Mục tiêu của bước này là làm mỏng cách cạnh biên bằng cách giữ lại các điểm cực đại của gradient và loại bỏ các điểm không phải là biên.
* Với mỗi điểm ảnh, kiểm tra các điểm lân cận theo hướng gradient(𝜃).
* Giữ lại nếu nó là điểm cực đại so với điểm lân cận.

A comparison of a person's face

Description automatically generated

Hình 6. Kết quả của thuật toán Non-maximum suppression[[8]](#footnote-8)[8]

* Bước 4: Ngưỡng kép.
* Ngưỡng kép giúp phân loại các điểm ảnh thành biên mạnh, biên yếu và không phải là biên.
* Áp dụng hai ngưỡng: ngưỡng cao (TH ​) và ngưỡng thấp (TL​).
  + Biên mạnh: điểm ảnh có giá trị gradient lớn hơn TH ​.
  + Biên yếu: điểm ảnh có giá trị gradient giữa TL và TH .
  + Không phải là biên: điểm ảnh có giá trị gradient nhỏ hơn TL ​.

A screenshot of a black and white image of a person

Description automatically generated

Hình 7. Hình ảnh trước non-maximum suppression (trái) – Kết quả (phải): pixel yếu có màu xám và pixel mạnh có màu trắng.[[9]](#footnote-9)[9]

* Bước 5: Theo dõi cạnh bằng độ trễ.
* Mục tiêu của bước này là liên kết các biên yếu với biên mạnh nếu chúng liền kề, đảm bảo tính liên tục của biên.
* Duyệt qua tất cả các điểm ảnh.
  + Nếu điểm ảnh là biên mạnh, giữ lại.
  + Nếu điểm ảnh là biên yếu và có ít nhất một điểm ảnh lân cận là biên mạnh, giữ lại.

A black and white image of a person

Description automatically generated

Hình 8[[10]](#footnote-10)[10]

* 1. Phương pháp lọc nhiễu.
* Thuật toán BFS.

1. Định nghĩa.

Tìm kiếm theo chiều rộng (BFS) là một thuật toán để duyệt đồ thị hoặc cây. BFS áp dụng cho cây và đồ thị gần như giống nhau. Sự khác biệt duy nhất là đồ thị có thể chứa các chu trình, vì vậy chúng ta có thể duyệt lại cùng một nút. Để tránh xử lý lại cùng một nút, chúng ta sử dụng mảng boolean đã truy cập, mảng này sẽ đánh dấu các đỉnh đã truy cập. BFS sử dụng cấu trúc dữ liệu hàng đợi (queue) để tìm đường đi ngắn nhất trong biểu đồ.

1. Tính chất.

* Đảm Bảo Liên Thông

BFS đảm bảo rằng tất cả các pixel thuộc về cùng một vùng liên thông sẽ được gán cùng một nhãn. Thuật toán này duyệt qua toàn bộ các pixel liên kề theo chiều rộng, đảm bảo không bỏ sót bất kỳ pixel nào trong vùng.

* Độ Phức Tạp Tính Toán

Độ phức tạp thời gian của BFS là O(V+E), trong đó V là số lượng pixel và E là số cạnh giữa các pixel. Trong trường hợp xử lý ảnh, E thường tỷ lệ với V, do đó, độ phức tạp thời gian là O(n⋅m)O, với nnn là chiều rộng và mmm là chiều cao của ảnh. Điều này đảm bảo BFS hoạt động hiệu quả trên các ảnh có kích thước lớn.

* Không Gian Bộ Nhớ

BFS sử dụng hàng đợi để lưu trữ các pixel chờ xử lý. Trong trường hợp xấu nhất, hàng đợi có thể chứa tất cả các pixel của ảnh, do đó không gian bộ nhớ cần thiết là O(n⋅m).

* Tính Chất Hàng Đợi

BFS sử dụng hàng đợi để xử lý các pixel theo thứ tự, đảm bảo rằng các pixel gần nhau được xử lý trước, giúp gán nhãn một cách hệ thống và chính xác.

* Tính Ổn Định

BFS đảm bảo tính ổn định trong việc phân loại vùng liên thông. Mỗi lần chạy thuật toán BFS với cùng một ảnh đầu vào sẽ cho ra cùng một kết quả phân loại vùng liên thông.

* Khả Năng Xử Lý Nhiều Thành Phần

BFS có thể xử lý nhiều thành phần liên thông trong một ảnh duy nhất. Mỗi khi phát hiện một pixel chưa được gán nhãn, thuật toán sẽ bắt đầu một cuộc tìm kiếm mới để gán nhãn cho toàn bộ vùng liên thông liên quan.

1. Giải thích thuật toán.

 **Khởi tạo**.

* Tạo một ma trận nhãn để đánh dấu các pixel đã được gán nhãn, tất cả các phần tử ban đầu bằng 0. Khởi tạo nhãn bắt đầu từ 1.

 **Duyệt qua từng pixel trong ảnh**.

* Nếu pixel chưa được gán nhãn và thuộc foreground, bắt đầu một cuộc tìm kiếm BFS từ pixel này.

 **BFS để gán nhãn cho vùng liên thông**.

* Sử dụng hàng đợi để duyệt các pixel lân cận và gán nhãn cho các pixel thuộc cùng một vùng liên thông.

 **Gán nhãn cho toàn bộ vùng liên thông**.

* Tiếp tục duyệt cho đến khi tất cả các pixel được gán nhãn.
  1. Phương pháp phân đoạn ảnh.

Thuật toán phân ngưỡng Otsu.

1. Định nghĩa.

Thuật toán Otsu là một phương pháp tự động xác định ngưỡng phân đoạn trong xử lý ảnh để tách một ảnh xám thành hai vùng: vật thể và nền ảnh. Thuật toán này dựa trên việc tối ưu hóa phương sai giữa các lớp (inter-class variance) để tìm ngưỡng tốt nhất phân biệt giữa hai vùng trong ảnh. Nobuyuki Otsu đã phát triển phương pháp này vào năm 1979 và nó đã trở thành một trong những kỹ thuật phân ngưỡng phổ biến nhất.

1. Tính chất.

* Tự Động Tìm Ngưỡng

Thuật toán Otsu tự động xác định ngưỡng tối ưu để phân tách các pixel trong ảnh thành hai lớp (foreground và background) mà không cần bất kỳ thông tin bên ngoài nào. Điều này làm cho nó trở thành một công cụ hữu ích cho nhiều ứng dụng xử lý ảnh.

* Dựa Trên Phương Sai Giữa Các Lớp

Thuật toán Otsu hoạt động dựa trên việc tối ưu hóa phương sai giữa các lớp. Nó tìm ngưỡng sao cho phương sai giữa các lớp (background và foreground) được tối đa hóa. Cụ thể:

* Phương sai tổng thể (​) của ảnh được tính bằng cách sử dụng toàn bộ các giá trị cường độ của ảnh.
* Phương sai trong lớp (​) đo lường sự phân tán trong mỗi lớp (background và foreground).
* Phương sai giữa các lớp (​) đo lường sự phân tán giữa hai lớp này và được tối đa hóa trong thuật toán Otsu.
* Không Yêu Cầu Các Tham Số Ban Đầu

Một trong những ưu điểm lớn của thuật toán Otsu là nó không yêu cầu bất kỳ tham số ban đầu nào từ người dùng. Ngưỡng tối ưu được tính toán trực tiếp từ histogram của ảnh.

* Hiệu Quả Tính Toán

Thuật toán Otsu tính toán ngưỡng tối ưu dựa trên histogram của ảnh, do đó nó rất hiệu quả về mặt tính toán. Việc tính toán này có thể được thực hiện một cách nhanh chóng và dễ dàng ngay cả với các ảnh có kích thước lớn.

* Áp Dụng Cho Các Ảnh Cấp Xám

Thuật toán Otsu được thiết kế chủ yếu cho các ảnh cấp xám (grayscale images). Đối với các ảnh màu, có thể cần áp dụng các biến thể của thuật toán hoặc chuyển đổi ảnh màu sang ảnh cấp xám trước khi áp dụng Otsu.

* Khả Năng Ứng Dụng Cho Phân Ngưỡng Đa Cấp

Mặc dù thuật toán Otsu gốc chỉ xác định một ngưỡng để phân chia ảnh thành hai lớp, nó có thể được mở rộng để xác định nhiều ngưỡng (multi-thresholding) nhằm phân chia ảnh thành nhiều lớp. Điều này được thực hiện bằng cách tối ưu hóa tổng phương sai giữa các lớp cho nhiều ngưỡng.

* Nhạy Cảm Với Histogram Của Ảnh

Hiệu suất của thuật toán Otsu phụ thuộc vào histogram của ảnh. Đối với các ảnh có histogram rõ ràng với hai đỉnh riêng biệt, thuật toán Otsu hoạt động rất tốt. Tuy nhiên, đối với các ảnh có histogram phẳng hoặc có nhiễu, kết quả có thể không tối ưu.

* Độ Ổn Định Cao

Thuật toán Otsu thường cho kết quả ổn định và nhất quán, đặc biệt là khi histogram của ảnh có sự phân biệt rõ ràng giữa các lớp.

1. Giải thích thuật toán.

* Bước 1: Tính histogram và xác suất xuất hiện của các mức xám
* Duyệt qua tất cả các pixel trong ảnh để tính histogram.
* Tính xác suất xuất hiện của mỗi mức xám P(i):

P(i) = (2.6)

trong đó ni​ là số lượng pixel có mức xám i và N là tổng số pixel trong ảnh.

* Bước 2: Khởi tạo các giá trị ban đầu
* Tính giá trị trung bình toàn cục của ảnh

μT = (2.7)

* Bước 3: Duyệt qua tất cả các cặp ngưỡng có thể (từ 0 đến 255)
* Với mỗi cặp ngưỡng t1​ và t2​ (với 0 ≤ t1 < t2 ≤ 255), chia ảnh thành ba lớp:
  + Lớp 1: [0, t1​]
  + Lớp 2: [t1 + 1, t2​]
  + Lớp 3: [t2 + 1, 255]
* Tính xác suất và giá trị trung bình của mỗi lớp:
  + Tổng xác suất của lớp 1:

ω0 = (2.8)

* + Tổng xác suất của lớp 2:

ω1 = (2.9)

* Tổng xác suất của lớp 3:

ω2 = (2.10)

* Giá trị trung bình của lớp 1:

μ0 = (2.11)

* Giá trị trung bình của lớp 2:

μ1 = (2.11)

* + Giá trị trung bình của lớp 3:

μ2 = (2.12)

* Bước 4: Tính tổng phương sai giữa các lớp
* Tính tổng phương sai giữa các lớp ​:

​ = ω0 . + ω1 . + ω2 . (2.13)

* Bước 5: Tìm ngưỡng tối ưu

Tìm cặp ngưỡng t1​ và t2​ làm cho tổng phương sai giữa các lớp ​ lớn nhất.

1. Thực hiện xây dựng hệ thống xử lí ảnh.

* Lưu đồ giải thuật.

A group of white rectangular objects with text

Description automatically generated

Hình 9. Lưu đồ giải thuật hệ thống xử lí ảnh.

* *Kết quả thực hiện xử lí ảnh theo quy trình ở phần trên.*

A collage of images of a diamond

Description automatically generated

Hình . Kết quả của quy trình từ 1 bước ảnh đầu vào có thể vẽ lại được biên dạng.

* *Mô tả cụ thể các bước*

Bước 1: Nhận dữ liệu từ camera:

* Mô tả: Camera chụp ảnh và truyền dữ liệu dưới dạng mảng một chiều chứa các giá trị điểm ảnh.
* Mục tiêu: Đảm bảo dữ liệu ảnh được nhận đầy đủ và chính xác để chuẩn bị cho các bước xử lý tiếp theo.

Bước 2: Chuyển đổi dữ liệu từ mảng 1 chiều thành mảng 2 chiều:

* Mô tả: Chuyển dữ liệu từ mảng một chiều thành mảng hai chiều với kích thước 640x480, đại diện cho bức ảnh xám.
* Mục tiêu: Tạo cấu trúc dữ liệu thích hợp để dễ dàng thực hiện các bước xử lý ảnh.
* Chương trình nhận dữ liệu và chuyển dữ liệu từ dạng 1 chiều sang dạng 2 chiều bằng ngôn ngữ C#:

// Dữ liệu nhận về từ Camera được lưu vào mảng 1 chiều

byte[] byteArrayModified = byteData.ToArray();

// Kích thước bức ảnh (chọn được từ kích thước bức ảnh của camera)

int width = 640; // Adjust according to your image width

int height = 480; // Adjust according to your image height

int newWidth = width;

int newHeight = height;

// chuyển từ dạng 1 chiều sang 2 chiều để mô tả 1 bức ảnh xám

byte[,] byteArray2D = new byte[newHeight, newWidth];

byteArray2D = ConvertTo2DArray(byteArrayModified, newHeight, newWidth);

//thể hiện bức ảnh đầu vào.

Image<Bgr, byte> Hinhmucxam = new Image<Bgr, byte>(newWidth, newHeight);

* Kết quả:

A square object in the dark

Description automatically generated

Hình 9. Kết quả thể hiện ảnh đầu vào.

Bước 3: Áp dụng thuật toán Otsu để phân biệt vật thể và nền:

* Mô tả: Thuật toán Otsu được sử dụng để tìm ngưỡng phân loại giữa vật thể và nền. Các điểm ảnh của vật thể được gán giá trị 255 (trắng), và nền được gán giá trị 0 (đen).
* Mục tiêu: Phân biệt rõ ràng giữa vật thể và nền trong ảnh.

Chương trình thuật toán Otsu được ghi rõ các bước thực hiện và công thức ở phần lý thuyết, sau đây là phần triển khai thuật toán bằng ngôn ngữ C#:

// Tính histogram bước ảnh với kích thước (width.height)

int[] histogram = new int[256];

for (int i = 0; i < width; i++)

{

for (int j = 0; j < height; j++)

{

histogram[grayImage[i, j]]++;

}

}

// Tính xác suất

double[] probability = new double[256];

for (int i = 0; i < 256; i++)

{

probability[i] = (double)histogram[i] / totalPixels;

}

// Tìm ngưỡng tối ưu với Ob min

double maxVariance = double.MinValue;

int threshold1 = 0;

int threshold2 = 0;

for (int t1 = 0; t1 < 256; t1++)

{

for (int t2 = t1 + 1; t2 < 256; t2++)

{

// Các lớp

double w0 = 0, w1 = 0, w2 = 0;

double m0 = 0, m1 = 0, m2 = 0;

for (int i = 0; i < t1; i++)

{

// Công thưc 2.8 , 2.9

w0 += probability[i];

m0 += i \* probability[i];

}

for (int i = t1; i < t2; i++)

{

// Công thưc 2.10 , 2.11

w1 += probability[i];

m1 += i \* probability[i];

}

for (int i = t2; i < 256; i++)

{

// Công thưc 2.12 , 2.13

w2 += probability[i];

m2 += i \* probability[i];

}

if (w0 > 0) m0 /= w0;

if (w1 > 0) m1 /= w1;

if (w2 > 0) m2 /= w2;

// Tìm ngưỡng tối ưu threshold1 và threshold2 để cho Ob min, công thức (2.13)

double betweenClassVariance = w0 \* w1 \* Math.Pow(m0 - m1, 2) + w1 \* w2 \* Math.Pow(m1 - m2, 2) + w0 \* w2 \* Math.Pow(m0 - m2, 2);

if (betweenClassVariance > maxVariance)

{

maxVariance = betweenClassVariance;

threshold1 = t1;

threshold2 = t2;

}

}

}

result[0] = threshold1;

result[1]= threshold2;

* Kết quả đạt được

A white square on a black background

Description automatically generated

Hình 10. Kết quả phân đoạn ảnh bằng ngưỡng Otsu.

Bước 4: Lọc các vùng vật thể nhỏ bằng thuật toán BFS:

* Mô tả: Loại bỏ các phần vật thể có diện tích nhỏ và chỉ giữ lại phần có diện tích lớn nhất.
* Mục tiêu: Đảm bảo chỉ các vật thể lớn nhất (có thể là mục tiêu chính) được giữ lại.
* Chương trình thuật toán BFS để đánh dấu các vùng trắng, sau đó chỉ lấy vùng lớn nhất:

private static int width;

private static int height;

// Hàm để kiểm tra xem tọa độ có nằm trong ảnh không

private static bool IsValid(int x, int y)

{

return x >= 0 && x < width && y >= 0 && y < height;

}

// Hàm để gán nhãn các vùng liên thông bằng thuật toán BFS

private static void LabelComponent(int[,] image, int[,] labels, int startX, int startY, int label)

{

int[] dx = { -1, 1, 0, 0 };

int[] dy = { 0, 0, -1, 1 };

Queue<(int, int)> queue = new Queue<(int, int)>();

queue.Enqueue((startX, startY));

labels[startX, startY] = label;

while (queue.Count > 0)

{

var (x, y) = queue.Dequeue();

for (int d = 0; d < 4; d++)

{

int newX = x + dx[d];

int newY = y + dy[d];

if (IsValid(newX, newY) && image[newX, newY] == 255 && labels[newX, newY] == 0)

{

labels[newX, newY] = label;

queue.Enqueue((newX, newY));

}

}

}

}

public static int[,] RemoveSmallWhiteRegions(int[,] image)

{

width = image.GetLength(0);

height = image.GetLength(1);

int[,] labels = new int[width, height];

int label = 1;

Dictionary<int, int> labelSizes = new Dictionary<int, int>();

// Gán nhãn cho các vùng liên thông

for (int x = 0; x < width; x++)

{

for (int y = 0; y < height; y++)

{

if (image[x, y] == 255 && labels[x, y] == 0)

{

LabelComponent(image, labels, x, y, label);

label++;

}

}

}

// Tính diện tích của mỗi vùng

for (int x = 0; x < width; x++)

{

for (int y = 0; y < height; y++)

{

if (labels[x, y] != 0)

{

if (!labelSizes.ContainsKey(labels[x, y]))

{

labelSizes[labels[x, y]] = 0;

}

labelSizes[labels[x, y]]++;

}

}

}

// Tìm nhãn của vùng lớn nhất

int maxLabel = -1;

int maxSize = -1;

foreach (var kvp in labelSizes)

{

if (kvp.Value > maxSize)

{

maxSize = kvp.Value;

maxLabel = kvp.Key;

}

}

// Tạo ảnh kết quả chỉ giữ lại vùng lớn nhất

int[,] result = new int[width, height];

for (int x = 0; x < width; x++)

{

for (int y = 0; y < height; y++)

{

if (labels[x, y] == maxLabel)

{

result[x, y] = 255;

}

else

{

result[x, y] = 0;

}

}

}

return result;

* Kết quả của thuật toán BFS

A white diamond shaped object on a black background

Description automatically generated

Hình 11. Sau khi lọc nhiễu bằng thuật toán BFS.

Bước 5: Phát hiện cạnh bằng thuật toán Canny:

* Mô tả: Áp dụng thuật toán Canny để phát hiện các cạnh của vật thể trong ảnh.
* Mục tiêu: Xác định các biên rõ ràng của vật thể trong ảnh.
* Bức ảnh đã được đưa về dạng binary (0 và 255) nên việc xác định ngưỡng TL và TH trở nên rất dễ dàng vì khi đạo hàm điểm ảnh chỉ còn 2 giá trị là 0 và 255. Chương trình mô tả các bước thực hiện của thuật toán Canny được bỏ qua bước làm mờ vì đã được làm mờ từ camera.

int[,] gradientMagnitude = new int[BlurredImage.GetLength(0), BlurredImage.GetLength(1)];

int[,] gradientAngle = new int[BlurredImage.GetLength(0), BlurredImage.GetLength(1)];

int[,] Result = new int[BlurredImage.GetLength(0), BlurredImage.GetLength(1)];

int[,] SobelX = {{ -1, 0, 1 },

{ -2, 0, 2 },

{ -1, 0, 1 }};

int[,] SobelY = {{ -1, -2, -1 },

{ 0, 0, 0 },

{ 1, 2, 1 }};

int white\_point = 255;

int gray\_point = 50;

int max\_gradient = 0;

//tính đạo hàm điểm ảnh và góc biến đổi của điểm ảnh dựa trên công thức 2.4 và 2.5

for (int x = 1; x < BlurredImage.GetLength(0) - 1; x++)

{

for (int y = 1; y < BlurredImage.GetLength(1) - 1; y++)

{

int sumX = 0;

int sumY = 0;

for (int i = -1; i <= 1; i++)

{

for (int j = -1; j <= 1; j++)

{

//tính đạo hàm điểm ảnh công thưc 2.2

sumX += (int)(SobelX[i + 1, j + 1] \* BlurredImage[x + i, y + j]);

//tính đạo hàm điểm ảnh công thưc 2.3

sumY += (int)(SobelY[i + 1, j + 1] \* BlurredImage[x + i, y + j]);

}

}

//tính đạo hàm điểm ảnh công thưc 2.4

gradientMagnitude[x, y] = (int)Math.Sqrt(sumX \* sumX + sumY \* sumY);

if (gradientMagnitude[x, y] > max\_gradient)

{

max\_gradient = gradientMagnitude[x, y];

}

if (gradientMagnitude[x, y] >= 255)

{

gradientMagnitude[x, y] = 255;

}

else if (gradientMagnitude[x, y] <= 0)

{

gradientMagnitude[x, y] = 0;

}

//tính đạo hàm điểm ảnh công thưc 2.5

gradientAngle[x, y] = (int)Math.Abs((Math.Atan2(sumY, sumX) \* 180 / Math.PI));

}

}

//thuật toán Non- maximum suppress.

for (int x = 1; x < BlurredImage.GetLength(0) - 1; x++)

{

for (int y = 1; y < BlurredImage.GetLength(1) - 1; y++)

{

double q = 255;

double r = 255;

if ((gradientAngle[x, y] >= 0 && gradientAngle[x, y] < 22.5) || (gradientAngle[x, y] <= 180 && gradientAngle[x, y] >= 157.5))

{

q = gradientMagnitude[x, y - 1];

r = gradientMagnitude[x, y + 1];

}

else if (gradientAngle[x, y] >= 22.5 && gradientAngle[x, y] < 67.5)

{

q = gradientMagnitude[x + 1, y - 1];

r = gradientMagnitude[x - 1, y + 1];

}

else if (gradientAngle[x, y] >= 67.5 && gradientAngle[x, y] < 112.5)

{

q = gradientMagnitude[x + 1, y];

r = gradientMagnitude[x - 1, y];

}

else if (gradientAngle[x, y] >= 112.5 && gradientAngle[x, y] < 157.5)

{

q = gradientMagnitude[x - 1, y - 1];

r = gradientMagnitude[x + 1, y + 1];

}

if (gradientMagnitude[x, y] >= q && gradientMagnitude[x, y] >= r)

{

Result[x, y] = gradientMagnitude[x, y];

}

else

{

Result[x, y] = 0;

}

if (Result[x, y] >= high\_threshold)

{

Result[x, y] = white\_point;

}

else if (Result[x, y] >= low\_threshold && Result[x, y] < high\_threshold)

{

Result[x, y] = gray\_point;

}

else

{

Result[x, y] = 0;

}

}

}

//xét điểm ảnh dựa trên ngưỡng TLvà TH

for (int x = 1; x < Result.GetLength(0) - 1; x++)

{

for (int y = 1; y < Result.GetLength(1) - 1; y++)

{

if (Result[x, y] == gray\_point)

{

if ((Result[x + 1, y - 1] == white\_point) ||

(Result[x + 1, y] == white\_point) ||

(Result[x + 1, y + 1] == white\_point) ||

(Result[x, y - 1] == white\_point) ||

(Result[x, y + 1] == white\_point) ||

(Result[x - 1, y - 1] == white\_point) ||

(Result[x - 1, y] == white\_point) ||

(Result[x - 1, y + 1] == white\_point))

{

Result[x, y] = white\_point;

}

else

Result[x, y] = 0;

}

}

}

for (int x = 0; x < Result.GetLength(0); x++)

{

for (int y = 0; y < Result.GetLength(1); y++)

{

if (Result[x, y] >= 255)

Result[x, y] = 255;

else if (Result[x, y] <= 0)

Result[x, y] = 0;

if ((y < 5) || (y > Result.GetLength(1) - 5))

Result[x, y] = 0;

if ((x < 5) || (x > Result.GetLength(0) - 5))

Result[x, y] = 0;

}

}

gradient = gradientMagnitude;

return Result;

A white outline of a diamond on a black background

Description automatically generated

Hình 12. Nhận diện cạnh bằng phương pháp Canny.

Bước 7: Vẽ ma trận Hough từ ảnh cạnh:

* Mô tả: Sử dụng phương pháp Hough Transform để vẽ ma trận Hough từ ảnh cạnh.
* Mục tiêu: Xác định các đường thẳng có trong ảnh.
* Chương trình vẽ sơ đồ Hough.

int width = image.GetLength(0);

int height = image.GetLength(1);

// Tính số cột của ma trận Hough dựa trên đường chéo dài nhất trong ảnh

int diagonalLength = (int)Math.Ceiling(Math.Sqrt(width \* width + height \* height));

int houghWidth = 180; // Số cột của ma trận Hough

int houghHeight = diagonalLength \* 2; // Số hàng của ma trận Hough

int[,] houghMatrix = new int[houghWidth, houghHeight];

// Thực hiện Hough Transform

for (int x = 0; x < width; x++)

{

for (int y = 0; y < height; y++)

{

if (image[x, y] > 0) // Nếu điểm ảnh là điểm biên

{

for (int theta = 0; theta < houghWidth; theta++)

{

double radian = (theta \* Math.PI) / 180.0;

int rho = (int)(x \* Math.Cos(radian) + y \* Math.Sin(radian));

rho += diagonalLength; // Dịch chuyển rho để không có giá trị âm

houghMatrix[theta, rho] = houghMatrix[theta, rho] + 1;

if (houghMatrix[theta, rho] > 255)

{

houghMatrix[theta, rho] = 255;

}

}

}

}

}

return houghMatrix;

A black background with red circles and lines

Description automatically generated

Hình 13. Ma trận Hough để chọn ra các cặp (r,θ)

Bước 8: Lấy các điểm đặc trưng từ ma trận Hough:

* Mô tả: Trích xuất các điểm đặc trưng từ ma trận Hough.
* Mục tiêu: Xác định các điểm quan trọng đại diện cho các đường thẳng trong ảnh.

A black background with red circles and white text

Description automatically generated

Bước 9: Quy đổi các điểm đặc trưng về phương trình đường thẳng và tìm giao điểm:

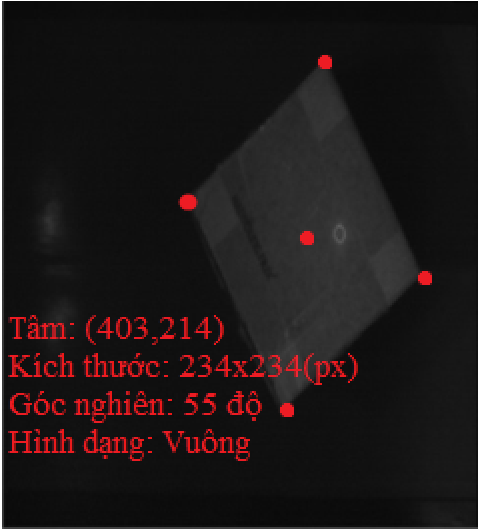
* Mô tả: Chuyển đổi các điểm đặc trưng về phương trình đường thẳng và tìm các giao điểm giữa các đường này.
* Mục tiêu: Xác định các giao điểm để nhận dạng hình dạng vật thể.

A black background with red dots

Description automatically generated

Bước 10: Phân loại hình dạng của ảnh:

* Mô tả: Dựa trên số lượng phương trình và giao điểm, tiến hành phân loại hình dạng của vật thể:
  + 3 phương trình và 3 giao điểm: Hình tam giác.
  + 4 phương trình, 4 giao điểm và 4 góc vuông: Hình chữ nhật hoặc hình vuông.
  + Nếu không thỏa điều kiện trên, kiểm tra xem có phải hình tròn hay không dựa trên khoảng cách từ tâm tới các điểm biên.
* Mục tiêu: Nhận dạng và phân loại hình dạng của vật thể trong ảnh.



1. Kết quả, đánh giá hệ thống.
   1. Kết quả nhận diện một số ảnh.

A screenshot of a computer

Description automatically generated

Hình 15. Nhận dạng hình hộp chữ nhật.

A screenshot of a computer screen

Description automatically generated

Hình 16. Nhận dạng hình hộp tam giác.

A screenshot of a computer

Description automatically generated

Hình 17. Nhận dạng hình tròn.

A screenshot of a computer

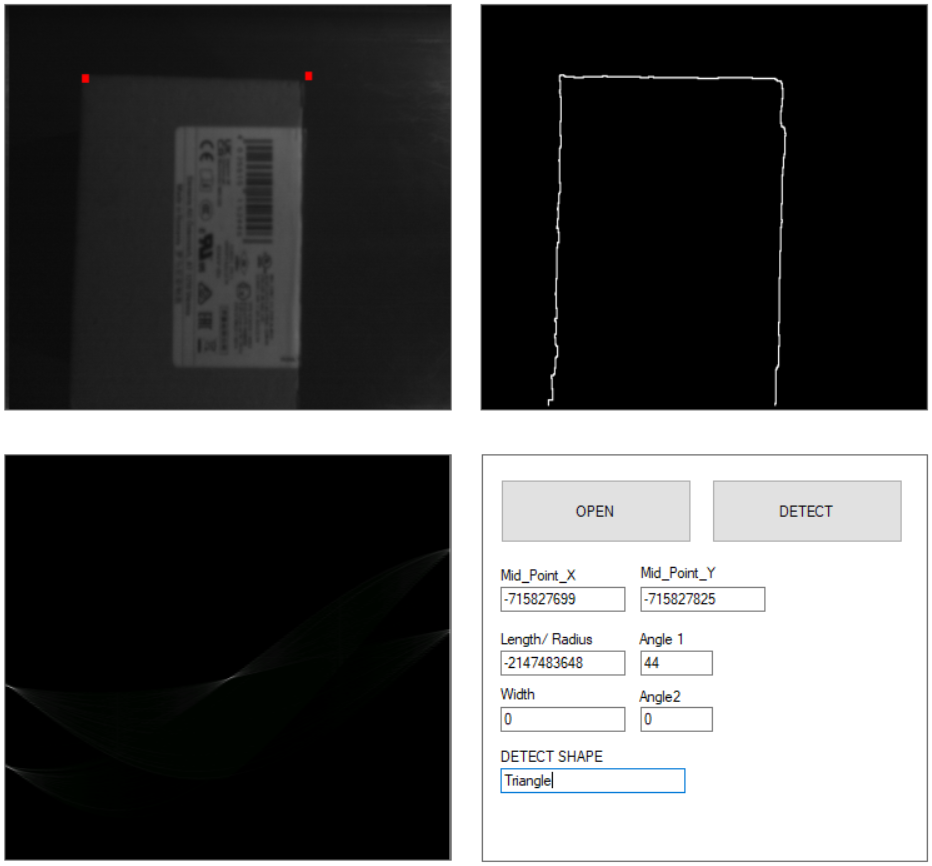
Description automatically generated

Hình 18. Nhận dạng hình vuông.

* 1. Thực hiện đánh giá thực tế.
* Xử lí liên tục 20 tấm ảnh cho ra kết quả.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Hình dạng | Chiều dài / bán kính (px) | Chiều rộng (px) | Góc hợp với Ox (độ) | Vị trí tâm (x,y) | Thời gian thực thi (s) | Đúng / Sai |
| 1 | Hình chữ nhật | 311 | 270 | 121 | (321,220) | 1.175 | Đúng |
| 2 | Hình chữ nhật | 313 | 272 | 114 | (432,250) | 1.1732 | Đúng |
| 3 | Hình tam giác | - | - | - | - | - | Sai |
| 4 | Hình chữ nhật | 433 | 377 | 91 | (305,291) | 1.1599 | Đúng |
| 5 | Hình chữ nhật | 432 | 376 | 126 | (325,323) | 1.099 | Đúng |
| 6 | Hình chữ nhật | 432 | 376 | 157 | (351,153) | 1.0203 | Đúng |
| 7 | Hình vuông | 229 | - | 45 | (390,196) | 1.233 | Đúng |
| 8 | Hình vuông | 227 | - | 81 | (321,227) | 1.242 | Đúng |
| 9 | Hình vuông | 300 | - | 72 | (300,229) | 1.137 | Đúng |
| 10 | Hình vuông | 301 | - | 44 | (321,300) | 1.328 | Đúng |
| 11 | Hình tam giác | 297 | - | 91 | (290,421) | 1.226 | Đúng |
| 12 | Hình tam giác | 355 | - | 126 | (321,222) | 1.337 | Đúng |
| 13 | Hình tam giác | 229 | - | 157 | (403,223) | 1.141 | Đúng |
| 14 | Hình tam giác | 227 | - | 45 | (425,220) | 1.245 | Đúng |
| 15 | Hình tam giác | 300 | - | 81 | (321,225) | 1.323 | Đúng |
| 16 | Hình tròn | 301 | - | - | (274,226) | 1.232 | Đúng |
| 17 | Hình tròn | 229 | - | - | (226,224) | 1.055 | Đúng |
| 18 | Hình tròn | 227 | - | - | (428,223) | 1.023 | Đúng |
| 19 | Hình tròn | 300 | - | - | (329,225) | 1.112 | Đúng |
| 20 | Hình tròn | 301 | - | - | (341,275) | 1.221 | Đúng |

* Trường hợp sai kết quả.



* Nguyên nhân: sai kết quả là do ảnh chụp bị khuất hoàn toàn một cạnh.
* Tổng kết đánh giá.
* Ưu điểm:
  + Hệ thống nhận diện có thể dự đoán được hình dạng vật thể, trọng tâm và kích thước (px) của vật thể.
  + Có thể nhận dạng được nếu vật thể bị khuất 1 phần của cạnh.
  + Độ sai lệch thấp.
* Nhược điểm:
  + Phụ thuộc vào độ sáng môi trường.
  + Hạn chế về phần nhận diện đường tròn.

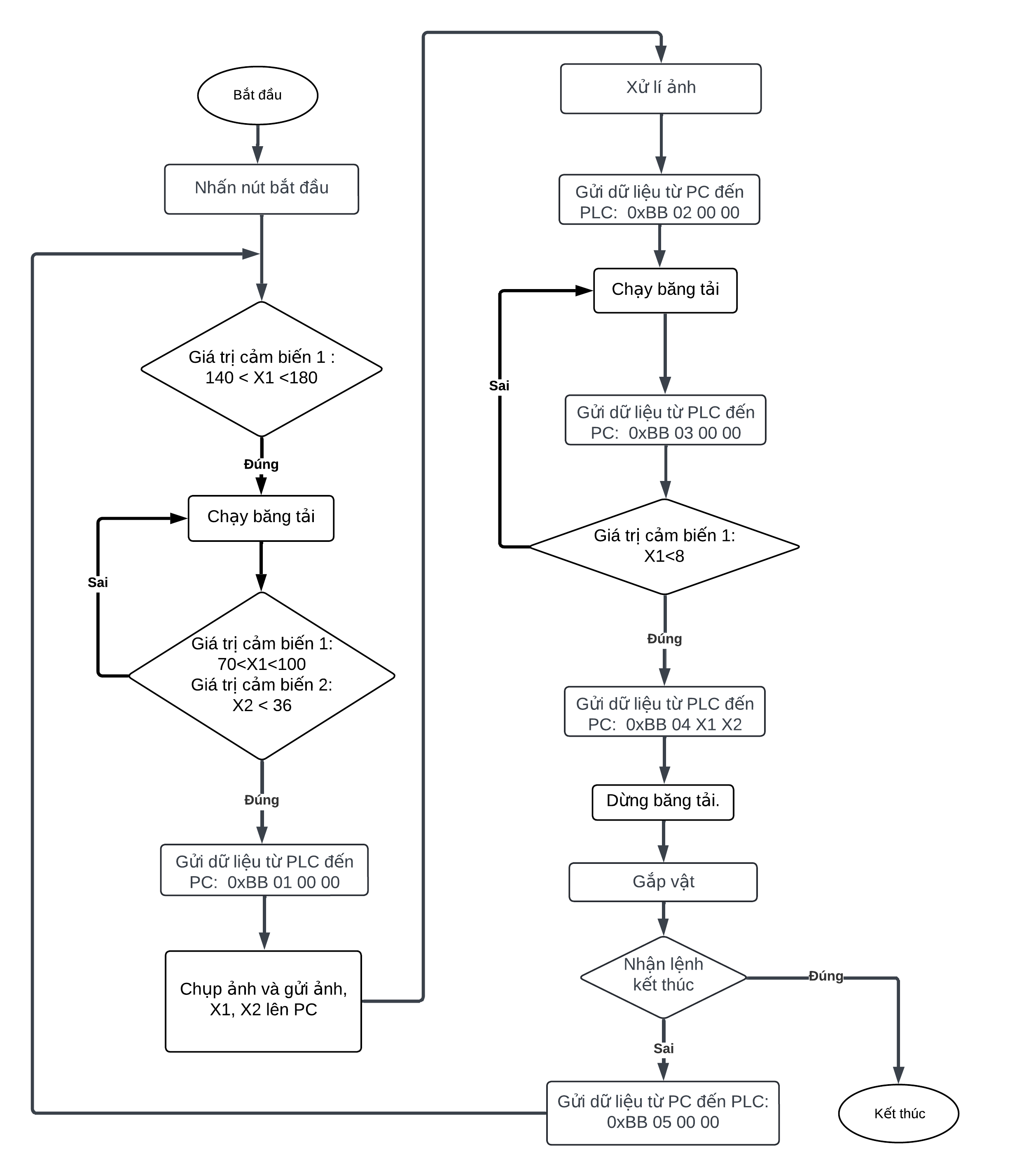
1. Phần điều khiển băng tải và lấy dữ liệu từ cảm biến.

* Sơ đồ kết nối.

Có file riêng trên `zalo

Thiết bị bao gồm:

* Module USART
* PLC S7-1214C
* Module IO
* Module điều khiển băng tải
* Module USART to TTL
* IO-Links AL1100
* Module GT68
* Sensor O5D150
* Lưu đồ điều khiển băng tải.



* Lấy dữ liệu từ cảm biến.

Cảm biến liên kết truyền nhận dữ liệu với PLC thông qua module IO-Link AL1100. Để lấy được dữ liệu trước hết phải khai báo module IO-Link AL1100 và tạo liên kết với PLC.

A screenshot of a computer

Description automatically generated

Khai báo địa chỉ của cảm biến trên module IO-Link AL1100, trong trường hợp này là Port 1 (IW68) và Port 3 (IW71).

A screenshot of a computer

Description automatically generated

Chuyển đổi dữ liệu từ dạng analog sang khoảng cách (cm).

A screenshot of a computer

Description automatically generated

Dựa vào datasheet của cảm biến O5D150 láy được

Với:

* IN1: Địa chỉ đầu vào của cảm biến
* IN2: Dữ liệu số bé nhất đo được.
* IN3: Dữ liệu số lớn nhất đo được.
* IN4: Khoản cách lớn nhất đo được.
* IN5: Khoản cách bé nhất đo được.
* IN6: sai số Offset của cảm biến.

Dựa vào datasheet của cảm biến O5D150 lấy các giá trị từ IN2 đến IN6:

IN2 = 65

IN3 = 3200

IN4 = 200

IN5 = 4

IN6 = 2.5

Công thức để quy đổi từ dữ liệu sang khoản cách.

D = )

1. [1] Quy trình phân tích ảnh chụp MR: https://www.researchgate.net/figure/Steps-involved-in-medical-image-processing-1-Source-image-2High-resolution-mean-MR\_fig2\_285989246 [↑](#footnote-ref-1)
2. [2] Hệ thống nhận diện khuông mặt: https://wesmart.vn/tin-tuc/cong-nghe-“ai”-nhan-dien-khuon-mat-va-nhung-tinh-nang-tuyet-voi-co-le-ban-chua-biet.html [↑](#footnote-ref-2)
3. [3] Barcode và QRcode trên sản phẩm: https://thigiacmaytinh.com/bai-viet-theo-chu-de/scan-barcode-qrcode/ [↑](#footnote-ref-3)
4. [4] Nhận diện biển số xe: www.futech.com.vn/tin-bai/tin-cong-ty/phat-hien-va-xu-ly-vi-pham-giao-thong-bang-hinh-anh/ [↑](#footnote-ref-4)
5. [5] Thiết bị xử lí ảnh VSTech: https://xulyanhcongnghiep.com/may-xu-ly-anh-cong-nghiep-cvs+4/product.html [↑](#footnote-ref-5)
6. [6] [↑](#footnote-ref-6)
7. [7] [↑](#footnote-ref-7)
8. [8] [↑](#footnote-ref-8)
9. [9] [↑](#footnote-ref-9)
10. [10] [↑](#footnote-ref-10)