HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
CƠ SỞ THÀNH PHỐ HỒ CHÍ MINH

KHOA CÔNG NGHỆ THÔNG TIN II

BÀI TẬP LỚN

MÔN: XỬ LÝ ẢNH

ĐỀ TÀI: MÔ PHỎNG NHẬN DIỆN MÃ VẠCH

Giảng viên hướng dẫn: Th.S Nguyễn Hữu Phong

Nhóm thực hiện: Nhóm 18

1. **Vũ Thị Hồng Oanh** **N19DCCN135**
2. **Nguyễn Quang Niên** **N19DCCN116**
3. **Đồng Ngọc Thủy Tiên** **N19DCCN167**

TP. HCM, 11/2022

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
CƠ SỞ THÀNH PHỐ HỒ CHÍ MINH

KHOA CÔNG NGHỆ THÔNG TIN II

BAI TẬP LỚN

MÔN: XỬ LÝ ẢNH

ĐỀ TÀI: MÔ PHỎNG NHẬN DIỆN MÃ VẠCH

Giảng viên hướng dẫn: Th.S Nguyễn Hữu Phong

Nhóm thực hiện: Nhóm 18

1. **Vũ Thị Hồng Oanh** **N19DCCN135**
2. **Nguyễn Quang Niên** **N19DCCN116**
3. **Đồng Ngọc Thủy Tiên** **N19DCCN167**

TP. HCM, 11/2022

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

LỜI CẢM ƠN

Lời đầu tiên, chúng em xin chân thành cảm ơn đến thầy Nguyễn Hữu Phong. Trong quá trình học tập và tìm hiểu môn Xử lý ảnh, nhờ nhận được sự hướng dẫn và giúp đỡ tận tình của thầy, chúng em đã có thể xây dựng đề tài mô phỏng nhận diện mã vạch một cách hoàn chỉnh, đồng thời tích lũy được một lượng kiến thức vô cùng cần thiết về bộ môn Xử lý ảnh nói riêng và ngành Công nghệ thông tin nói chung.

Chúng em xin cảm ơn các đơn vị, các anh chị và các bạn đã tham gia khảo sát để chúng em có thể hoàn thành đề tài mô phỏng nhận diện mã vạch có tính ứng dụng cao trong thực tế.

Tuy nhiên, với sự thiếu sót về kiến thức và kinh nghiệm cũng như thời gian có hạn nên khó tránh những hạn chế, chúng em mong nhận được sự đóng góp ý kiến của thầy để chúng em có thể học hỏi thêm được những kiến thức và kinh nghiệm giúp ích cho quá trình học tập và công việc sau này.

Chúng em xin trân trọng cảm ơn!

Thành phố Hồ Chí Minh, tháng 10 năm 2022

Nhóm sinh viên thực hiện

MỤC LỤC

Phần 1. GIỚI THIỆU

1.1. Lý do chọn đề tài:

Kể từ khi mã vạch được sử dụng thương mại lần đầu tiên vào năm 1966, vẫn cần phải có một số phương pháp tiêu chuẩn để làm cho hệ thống này được chấp nhận bởi toàn bộ nền công nghiệp. Năm 1970, tập đoàn Logion đã phát triển Mã nhận dạng thực phẩm chung (Universal Food Identification Code). Công ty đầu tiên sản xuất thiết bị mã vạch để bán lẻ (UGPIC) là một công ty của Mỹ. Sau đó, mã vạch lần đầu tiên được tập đoàn Tippecanoe Systems giới thiệu vào năm 1991. Kể từ đó, mã vạch đã trở thành một trong những ứng dụng phần mềm mã vạch phổ biến nhất cho phép các doanh nghiệp nhỏ áp dụng mã vạch mà không cần tốn nhiều công sức.

Nhận dạng mã vạch cần thiết trong nhiều ứng dụng trong nền công nghiệp hiện đại trên thế giới. Tất cả các mã vạch là một dạng truyền dữ liệu trực quan, có thể đọc được bằng máy thường mô tả một số dữ liệu về đối tượng mang mã vạch. Chúng ta biết rằng ban đầu mã vạch chỉ được quét bằng máy quét quang học đặc biệt được gọi là máy đọc mã vạch, mặc dù chúng ta đã phát hiện ra mã vạch mà không cần sử dụng đầu đọc mã vạch. Sau đó, phần mềm ứng dụng đã có sẵn cho các thiết bị có thể đọc hình ảnh, chẳng hạn như điện thoại thông minh có camera. Hệ thống nhận dạng mã vạch sử dụng xử lý hình ảnh hiện đã được phát triển.

Để kiểm soát thực tế việc sản xuất, lưu trữ và phân phối hàng hóa, trong khi việc này là cần thiết để cung cấp cho mỗi mặt hàng thông tin đã được sàng lọc, để xác định chính xác và lưu trữ thông tin này trong máy tính. Chúng ta tin rằng việc áp dụng hệ thống mã vạch có thể được sử dụng để coi hàng hóa là dữ liệu giúp thực hiện các quy trình khác nhau, bao gồm lưu trữ, phân loại, sản xuất và phân phối. Hệ thống nhanh nhạy với sự nhận dạng cơ học của từng mã vạch này là rất cần thiết trong các cơ sở sản xuất lớn hoặc các trung tâm phân phối lớn, nơi có số lượng lớn sản phẩm đã bị trộn lẫn.

Mã vạch đóng vai trò rất quan trọng trong việc bán hàng và quản lý hàng hóa của các doanh nghiệp. Nó đem lại rất nhiều lợi ích cho các doanh nghiệp cụ thể là: tiết kiệm thời gian, chi phí, có tính chính xác cao, kiểm soát được thông tin của sản phẩm, thuận lợi trong việc trao đổi thông tin điện tử, nâng cao năng suất và hiệu quả bán hàng. Ngoài ra nó còn giúp các cơ quan nhà nước dễ dàng trong việc quản lý sản phẩm và hoạt động kinh doanh của doanh nghiệp đó. Tất cả những lợi ích trên được mang lại chỉ bởi một loại mã hiệu đặc biệt - mã vạch. Cho nên việc nhận diện mã vạch là vô cùng quan trọng và cần thiết với trong việc mua bán, giúp người mua và bán tiết kiệm thời gian trong việc trao đổi và quản lý hàng hóa, nhận biết được hàng thật hay giả, từ đó đem lại trải nghiệm

về sản phẩm tốt hơn. Vậy nên chúng em chọn đề tài "Mô phỏng nhận diện mã vạch làm đề tài nghiên cứu".

Phần 2. PHƯƠNG PHÁP NGHIÊN CỨU

2.1. Cơ sở lý thuyết:

2.1.1 Mã vạch là gì?

Mã số mã vạch là một trong các công nghệ nhận dạng và thu thập dữ liệu tự động các đối tượng là sản phẩm, dịch vụ, tổ chức hoặc địa điểm…dựa trên việc ấn định một mã số (hoặc chữ số) cho đối tượng cần phân định và thể hiện mã đó dưới dạng mã vạch để thiết bị (máy quét) có thể đọc được.

Phân loại mã vạch:

Có hai loại mã vạch: mã vạch 1D và mã vạch 2D. Tuy nhiên đề tài này chọn đối tượng là mã vạch 1D nên về ta sẽ tập trung vào phân biệt một số mã vạch 1D thông dụng.

+ *Mã vạch UPC (Universal Product Code)* là một dạng thức kí hiệu được mã hóa sử dụng phổ biến tại Mỹ, Canada,... Loại mã vạch này là chuỗi 11 số (có giá trị từ 0-9) và có một số kiểm tra ở cuối để tạo thành một chuỗi mã vạch hoàn chỉnh 12 số. UPC gồm 2 phần: phần mã vạch và phần số. UPC gồm nhiều biến thể như: UPC-A, UPC-B, UPC-C, UPC-2, UPC-5,... Đây là loại mã vạch thông dụng trong kinh doanh bán lẻ, siêu thị, hàng tiêu dùng, công nghiệp thực phẩm.
+ *Mã vạch EAN (European Article Number)* là dạng thức ký hiệu có hình thức tương tự như UPC với phần mã vạch và phần số. Mã EAN được sử dụng phổ biến tại nhiều quốc gia trên thế giới. EAN được cấu tạo bởi 4 nhóm gồm: mã quốc gia, mã doanh nghiệp, mã sản phẩm và số kiểm tra. Chính vì thế loại mã vạch này được sử dụng cho những sản phẩm lưu thông trên toàn cầu.
+ *Mã vạch Code 39* là loại hình mã vạch cho phép hiển thị cả chữ cái, chữ số và một vài ký hiệu đặc biệt (tối đa 39 ký tự) để biểu thị thông tin của sản phẩm. Bởi thế nên nó có thể chứa được lượng thông tin nhiều hơn UPC và EAN. Loại mã vạch này thông dụng trong hậu cần để mã hóa các định danh cụ thể từng ứng dụng hay được sử dụng bởi một số dịch vụ bưu chính.
+ *Mã vạch code 128* là một ký hiệu tuyến tính mật độ cao, mã hóa văn bản, số nhiều hàm và toàn bộ ký tự ASCII. Nó được ứng dụng phổ biến vì có nhiều ưu điểm như nhỏ gọn, lưu trữ thông tin đa dạng…. Nó được sử dụng rất nhiều trong ngành công nghiệp đóng tàu và đóng gói để xác định mức thùng chứa và pallet trong chuỗi cung ứng, mã hóa một lượng lớn dữ liệu trong một không gian nhỏ..

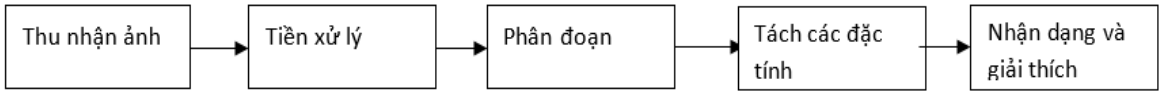
- **Cấu tạo của mã vạch**

Mã số mã vạch gồm có hai phần:

+ *Mã số*: Là một dãy số chữ nguyên, trong đó các nhóm số để chứng minh về xuất xứ hàng hóa. Do cách đánh số như vậy, mỗi loại hàng hóa sẽ có dãy số duy nhất để nhận dạng đơn nhất trên toàn thế giới. Đây là một cấu trúc mã số tiêu chuẩn dùng để nhận dạng sản phẩm hàng hóa trên các quốc gia, trên mỗi vùng lãnh thổ khác nhau, tương tự như cấu trúc mã số điện thoại để liên lạc quốc tế.
+ *Mã vạch*: là một dãy các vạch và khoảng trống song song xen kẽ được thiết kế theo một nguyên tắc nhất định để thể hiện mã số hoặc cả chữ lẫn số dưới dạng các thiết bị đọc có đầu gắn laser (máy quét) nhận dạng và đọc được còn gọi là thiết bị quang học.

2.1.2 Xử lý ảnh trong Matlab

- Xử lý ảnh là đối tượng nghiên cứu của lĩnh vực thị giác máy, là quá trình biến đổi từ một ảnh ban đầu sang một ảnh mới với các đặc tính và tuân theo ý muốn của người sử dụng. Xử lý ảnh có thể gồm quá trình phân tích, phân lớp các đối tượng, làm tăng chất lượng, phân đoạn và tách cạnh, gán nhãn cho vùng hay quá trình biên dịch các thông tin hình ảnh của ảnh.
- Cũng như xử lý dữ liệu bằng đồ hoạ, xử lý ảnh số là một lĩnh vực của tin học ứng dụng. Xử lý dữ liệu bằng đồ họa đề cập đến những ảnh nhân tạo, các ảnh này được xem xét như là một cấu trúc dữ liệu và được tạo bởi các chương trình. Xử lý ảnh số bao gồm các phương pháp và kỹ thuật biến đổi, để truyền tải hoặc mã hoá các ảnh tự nhiên.
- Mục đích của xử lý ảnh gồm:
  + Biến đổi ảnh làm tăng chất lượng ảnh.
  + Tự động nhận dạng ảnh, đoán nhận ảnh, đánh giá các nội dung của ảnh.
- Các quá trình xử lý ảnh:



+ *Thu nhận ảnh*: Ảnh đầu vào sẽ được thu nhận qua các thiết bị như camera, sensor, máy scanner,… và sau đó các tín hiệu này sẽ được số hóa. Việc lựa chọn các thiết bị thu nhận ảnh sẽ phụ thuộc vào đặc tính của các đối tượng cần xử lý. Các thông số quan trọng ở bước này là độ phân giải, chất lượng màu, dung lượng bộ nhớ và tốc độ thu nhận ảnh của các thiết bị.
+ *Tiền xử lý*: Ở bước này, ảnh sẽ được cải thiện về độ tương phản, khử nhiễu, khử bóng, khử độ lệch,… với mục đích làm cho chất lượng ảnh trở lên tốt hơn nữa, chuẩn bị cho

các bước xử lý phức tạp hơn về sau trong quá trình xử lý ảnh. Quá trình này thượng được thực hiện bởi các bộ lọc.

+ *Phân đoạn ảnh*: phân đoạn ảnh là bước then chốt trong xử lý ảnh. Giai đoạn này phân tích ảnh thành những thành phần có cùng tính chất nào đó dựa theo biên hay các vùng liên thông. Tiêu chuẩn để xác định các vùng liên thông có thể là cùng màu, cùng mức xám,… Mục đích của phân đoạn ảnh là để có một miêu tả tổng hợp về nhiều phần tử khác nhau cấu tạo lên ảnh thô. Vì lượng thông tin chứa trong ảnh rất lớn, trong khi đa số các ứng dụng chúng ta chỉ cần trích một vài đặc trưng nào đó, do vậy cần có một quá trình để giảm lượng thông tin khổng lồ đó. Quá trình này bao gồm phân vùng ảnh và trích chọn đặc tính chủ yếu.
+ *Tách các đặc tính*: Kết quả của bước phân đoạn ảnh thường được cho dưới dạng dữ liệu điểm ảnh thô, trong đó hàm chứa biên của một vùng ảnh, hoặc tập hợp tất cả các điểm ảnh thuộc về chính vùng ảnh đó. Trong cả hai trường hợp, sự chuyển đổi dữ liệu thô này thành một dạng thích hợp hơn cho việc xử lý trong máy tính là rất cần thiết. Để chuyển đổi chúng, câu hỏi đầu tiên cần phải trả lời là nên biểu diễn một vùng ảnh dưới dạng biên hay dưới dạng một vùng hoàn chỉnh gồm tất cả những điểm ảnh thuộc về nó. Biểu diễn dạng biên cho một vùng phù hợp với những ứng dụng chỉ quan tâm chủ yếu đến các đặc trưng hình dạng bên ngoài của đối tượng, ví dụ như các góc cạnh và điểm uốn trên biên chẳng hạn. Biểu diễn dạng vùng lại thích hợp cho những ứng dụng khai thác các tính chất bên trong của đối tượng, ví dụ như vân ảnh hoặc cấu trúc xương của nó. Sự chọn lựa cách biểu diễn thích hợp cho một vùng ảnh chỉ mới là một phần trong việc chuyển đổi dữ liệu ảnh thô sang một dạng thích hợp hơn cho các xử lý về sau. Chúng ta còn phải đưa ra một phương pháp mô tả dữ liệu đã được chuyển đổi đó sao cho những tính chất cần quan tâm đến sẽ được làm nổi bật lên, thuận tiện cho việc xử lý chúng.
+ *Nhận dạng và giải thích:* Đây là bước cuối cùng trong quá trình xử lý ảnh. Nhận dạng ảnh có thể được nhìn nhận một cách đơn giản là việc gán nhãn cho các đối tượng trong ảnh. Ví dụ đối với nhận dạng chữ viết, các đối tượng trong ảnh cần nhận dạng là các mẫu chữ, ta cần tách riêng các mẫu chữ đó ra và tìm cách gán đúng các ký tự của bảng chữ cái tượng ứng cho các mẫu chữ thu được trong ảnh. Giải thích là công đoạn gán nghĩa cho một tập các đối tượng đã được nhận biết.

- Các kiểu ảnh trong Matlab

+ *Ảnh được định chỉ số (Indexed Images)*
  - Một ảnh chỉ số bao gồm một ma trận dữ liệu X và ma trận bản đồ màu map. Ma trận dữ liệu có thể có kiểu thuộc lớp uint8, uint16 hoặc kiểu double. Ma trận bản đồ màu là một mảng mx3 kiểu double bao gồm các giá trị dấu phẩy động nằm

giữa 0 và 1. Mỗi hàng của bản đồ chỉ ra các giá trị red, green và blue của một màu đơn. Một ảnh chỉ số sử dụng ánh xạ trực tiếp giữa giá trị của pixel ảnh tới giá trị trong bản đồ màu. Màu sắc của mỗi pixel ảnh được tính toán bằng cách sử dụng giá trị tương ứng của X ánh xạ tới một giá trị chỉ số của map. Giá trị 1 chỉ ra hàng đầu tiên, giá trị 2 chỉ ra hàng thứ hai trong bản đồ màu …

- Một bản đồ màu thường được chứa cùng với ảnh chỉ số và được tự động nạp cùng với ảnh khi sử dụng hàm imread để đọc ảnh. Tuy nhiên, không bị giới hạn khi sử dụng bản đồ màu mặc định, có thể sử dụng bất kỳ bản đồ màu nào.

+ *Ảnh cường độ (Intensity Images)*

- Một ảnh cường độ là một ma trận dữ liệu ảnh I mà giá trị của nó đại diện cho cường độ trong một số vùng nào đó của ảnh. Matlab chứa một ảnh cường độ như một ma trận đơn, với mỗi phần tử của ma trận tương ứng với một pixel của ảnh. Ma trận có thể thuộc lớp double, uint8 hay uint16. Trong khi ảnh cường độ hiếm khi được lưu với bản đồ màu, Matlab sử dụng bản đồ màu để hiển thị chúng.
- Những phần tử trong ma trận cường độ đại diện cho các cường độ khác nhau hoặc độ xám. Những điểm có cường độ bằng 0 thường được đại diện bằng màu đen và cường độ 1,255 hoặc 65535 thường đại diện cho cường độ cao nhất hay màu trắng.

+ *Ảnh nhị phân (Binary Images)*

- Trong một ảnh nhị phân, mỗi pixel chỉ có thể chứa một trong hai giá trị nhị phân 0 hoặc 1. Hai giá trị này tương ứng với bật hoặc tắt (on hoặc off). Một ảnh nhị
- phân được lưu trữ như một mảng logic của 0 và 1.

+ *Ảnh RGB (RGB Images)*

- Một ảnh RGB – thường được gọi là true-color, được lưu trữ trong Matlab dưới dạng một mảng dữ liệu có kích thước 3 chiều mxnx3 định nghĩa các giá trị màu red, green và blue cho mỗi pixel riêng biệt. Màu của mỗi pixel được quyết định bởi sự kết hợp giữa các giá trị R, G, B (Red, Green, Blue) được lưu trữ trong một mặt phẳng màu tại vị trí của pixel. Định dạng file đồ hoạ lưu trữ ảnh RGB giống như một ảnh 24 bits trong đó R, G, B chiếm tương ứng 8 bit một. Điều này cho phép nhận được 16 triệu màu khác nhau.
- Một mảng RGB có thể thuộc lớp double, uint8 hoặc uint16. Trong một mảng RGB thuộc lớp double, mỗi thành phần màu có giá trị giữa 0 và 1. Một pixel mà thành phần màu của nó là (0, 0, 0) được hiển thị với màu đen và một pixel mà thành phần màu là (1, 1, 1) được hiển thị với màu trắng. Ba thành phần màu của mỗi pixel được lưu trữ cùng với chiều thứ 3 của mảng dữ liệu. Chẳng hạn, giá trị màu

R, G, B của pixel (10, 5) được lưu trữ trong RGB(10, 5, 1), RGB(10, 5, 2) và RGB(10, 5, 3) tương ứng.

- Để tính toán màu sắc của pixel tại hàng 2 và cột 3 chẳng hạn, ta nhìn vào bộ ba giá trị được lưu trữ trong (2, 3, 1:3). Giả sử (2, 3, 1) chứa giá trị 0.5176; (2, 3, 2) chứa giá trị 0.1608 và (2, 3, 3) chứa giá trị 0.0627 thì màu sắc của pixel tại (2, 3) sẽ là (0.5176, 0.1608, 0.0627).

2.1.3 Các hàm xử lý ảnh trong Matlab

- *Đọc ảnh đồ họa*

+ Hàm **imread** đọc các file ảnh với bất kỳ các định dạng ảnh đã biết hiện nay và lưu lại dưới dạng một ma trận biểu diễn ảnh trong MATLAB. Hầu hết các định dạng ảnh hiện nay dùng 8 bit cho mỗi pixel (ứng với một thành phần màu), do đó sau khi đọc MATLAB sẽ lưu lại dưới dạng ma trận thuộc kiểu uint8. Với các định dạng ảnh 16 bit như PNG và TIFF, MATLAB sẽ dùng kiểu uint16. Với ảnh index, ma trận màu sẽ được lưu với kiểu double. Cú pháp của hàm **imread**:

>> A = imread(filename,fmt)

>> [X,map] = imread(filename,fmt)

Trong đó: filename là chuỗi xác định tên file cần đọc cùng với đường dẫn (nếu file này không nằm trong thư mục hiện hành.

fmt là chuỗi cho biết định dạng của ảnh, thí dụ ‘bmp’, ‘gif’, ‘jpg’, …

Ngoài ra, hàm **imread** còn có thêm một số thông số khác tuỳ vào từng định dạng ảnh cụ thể. Gõ lệnh help ở cửa sổ lệnh của MATLAB để biết về các thông số này.

MATLAB hỗ trợ cả một số định dạng ảnh có thể chứa nhiều ảnh như HDF, TIFF hay GIF. Trong trường hợp mặc định, hàm imread chỉ đọc ảnh đầu tiên trong các ảnh này. Tuy nhiên, ta có thể cung cấp thêm một thông số là chỉ số của ảnh cần đọc trong dãy như trong ví dụ sau đây.

Ví dụ. Đọc một chuỗi 27 ảnh liên tiếp trong một file TIFF và lưu vào một dãy 4 chiều:

```
mri = uint8(zeros(128,128,1,27)); % khởi tạo một dãy 4 chiều

for frame=1:27
```

[mri(:,:,:,frame),map] = imread('mri.tif',frame);

End

+ Ghi đồ họa

Hàm **imwrite** cho phép lưu một ảnh biểu diễn bằng một ma trận trong MATLAB thành một file ảnh dưới một trong các định dạng ảnh đã biết. Cú pháp cơ bản của hàm này như sau:

>> imwrite (A, filename, fmt)

>> imwrite (X, map, filename, fmt)

>> imwrite (..., param1, val1, param2, val2, ...)

Có thể bỏ qua thông số fmt nếu trong chuỗi filename có cả phần mở rộng (sau dấu chấm). Tuỳ thuộc vào định dạng ảnh cần lưu, ta cung cấp thêm tên các thông số param1, param2, ... cùng với giá trị tương ứng của chúng. Ví dụ, lệnh sau đây thực hiện ghi vào file mypicture.jpg với chất lượng nén là 100:

>> imwrite(A, 'mypicture.jpg', 'Quality', 100)

Để kiểm chứng xem các thông số mà chúng ta đã xác định trong hàm **imwrite** có được thực hiện đúng hay không, hoặc để xem các thông số của một file ảnh nào đó, ta có thể dùng hàm **imfinfo**:

>> info = imfinfo(filename,fmt)

Ví dụ:

>> info = imfinfo('test.tif');

>> info.BitDepth ans =1

+ Các phép toán số học cơ bản

Các phép toán số học cơ bản trên các dữ liệu ảnh bao gồm các phép cộng, trừ, nhân và chia. Đây là những thao tác xử lý ảnh cơ bản trước khi thực hiện các biến đổi phức tạp

khác. Người sử dụng có thể dùng các phép toán số học mà MATLAB cung cấp để tác động lên dữ liệu ảnh. Tuy nhiên, do MATLAB chỉ hỗ trợ các phép toán này trên kiểu double nên cần thực hiện chuyển đổi kiểu trước khi thực hiện. Để làm giảm bớt thao tác này, trong MATLAB Image Processing Toolbox có cung cấp các hàm thực hiện các phép toán số học trên ảnh mà có thể chấp nhận bất kỳ kiểu dữ liệu ảnh nào và trả về kết quả thuộc cùng kiểu với các toán hạng. Các hàm này cũng xử lý các dữ liệu tràn một cách tự động. Dưới đây là danh sách các hàm thực hiện các phép toán số học cơ bản trên ảnh cùng với cú pháp tương ứng:

Tên hàm	**Cú pháp**	**Mô tả**
imabsdiff	z = imabsdiff(x,y)	Trừ mỗi phần tử của Y từ phần tử tương ứng của X, sau đó trả về trị tuyệt đối của hiệu
imadd	z = imadd(x,y,out_class)	Cộng hai ảnh hoặc cộng một ảnh với một hằng số, output_class là chuỗi xác định kiểu dữ liệu của tổng
imcomplement	im2 = imcomplement(im)	Lấy bù của ảnh im
imdivide	z = imdivide(x,y)	Chia các phần tử của ảnh x cho phần tử tương ứng của y, các giá trị phân số được làm tròn

imlincomb	z = imlincomb(k1,a1,k2,a2, ..., kn,an,k,out_class)	Lấy tổ hợp tuyến tính của các ảnh: z = k1.*a1 + k2.*a2 + ... + kn.*an + k
immultiply	z = immultiply(x,y)	Nhân hai ảnh hoặc nhân một ảnh với một hằng số, nếu kết quả bị tràn thì sẽ được giới hạn lại trong tầm cho phép
imsubstract	z = imsubtract(x,y)	Trừ hai ảnh hoặc trừ một ảnh cho một hằng số, nếu kết quả bị tràn thì sẽ được giới hạn lại trong tầm cho phép

- ✓ Phép cộng hai ảnh thường dùng để xếp chồng một ảnh lên trên một ảnh khác. Phép cộng một ảnh với một hằng số làm tăng độ sáng của ảnh.
- ✓ Phép trừ hai ảnh thường dùng để phát hiện những sự khác nhau giữa các ảnh trong một chuỗi các ảnh của cùng một cảnh. Phép trừ một ảnh cho một hằng số làm giảm độ sáng của ảnh.
- ✓ Phép nhân một ảnh với một hằng số, còn gọi là scaling, là một phép toán thường gặp trong xử lý ảnh. Nó làm tăng hoặc giảm độ sáng của ảnh tuỳ theo hệ số nhân là lớn hơn hay nhỏ hơn 1, nhưng khác với phép cộng ảnh với hằng số, phép nhân cho phép bảo toàn độ tương phản của ảnh, do đó ảnh có vẻ thực hơn.
- ✓ Phép chia hai ảnh cũng dùng để phát hiện những thay đổi giữa các ảnh liên tiếp của cùng một đối tượng nhưng dưới dạng những thay đổi tỷ lệ.

- *Các hàm hiển thị hình ảnh*

Để phục vụ chức năng hiển thị hình ảnh, MATLAB cung cấp hai hàm cơ bản là **image** và **imagesc**. Ngoài ra, trong Image Processing Toolbox cũng có hai hàm hiển thị ảnh khác, đó là **imview** và **imshow**.

- ✓ Hàm image(X, Y, C) hiển thị hình ảnh biểu diễn bởi ma trận C kích thước MxN lên trục toạ độ hiện hành. X, Y là các vector xác định vị trí của các pixel C(1,1) và C(M,N) trong hệ trục hiện hành. Toạ độ của pixel C(1,1) là (X(1),Y(1)) còn toạ độ của pixel C(M,N) là (X(end),Y(end)). Nếu không cung cấp X, Y, thì MATLAB sẽ mặc định là toạ độ của C(1,1) là (1,1), của C(M,N) là (M,N). Nếu ma trận C chỉ có hai chiều thì MATLAB hiểu rằng đây là ma trận ảnh dạng index với ma trận màu là ma trận màu hiện hành trong hệ thống. Ngoài ra có thể cung cấp thêm các cặp thông số (tên thuộc tính/ giá trị thuộc tính) cho hàm **image**. Ví dụ, hàm image(...,'parent',ax) xác định hệ trục toạ độ mà ảnh sẽ hiển thị trên đó là hệ trục toạ độ ax.
- ✓ Hàm **imagesc** có chức năng tương tự như hàm image, ngoại trừ việc dữ liệu ảnh sẽ được co giãn (scale) để sử dụng toàn bộ bản đồ màu hiện hành.
- ✓ Hàm **imview** cho phép hiển thị hình ảnh trên một cửa sổ riêng, nền Java, gọi là Image Viewer. Image Viewer cung cấp các công cụ cho phép dò tìm và xác định giá trị các pixel một cách linh hoạt. Sử dụng hàm này khi ta cần khảo sát bức ảnh và cần các thông tin về các pixel.
- ✓ Giống như các hàm **image** và **imagesc**, hàm **imshow** cũng tạo một đối tượng đồ hoạ thuộc loại image và hiển thị ảnh trên một figure. Hàm **imshow** sẽ tự động thiết lập các giá trị của các đối tượng image, axes và figure để thể hiện hình ảnh. Sử dụng hàm này trong các trường hợp ta cần lợi dụng các công cụ chú giải và các hỗ trợ in ấn có sẵn trong figure.

- Các phép biến đổi hình học:

  Phép nội suy ảnh:

- ✓ Nội suy là quá trình ước lượng giá trị của ảnh tại một điểm nằm giữa hai pixel có giá trị đã biết. Chẳng hạn, nếu ta thay đổi kích thước ảnh sao cho nó chứa nhiều pixel hơn ảnh gốc, thì giá trị của các pixel thêm vào sẽ được xác định bằng phép nội suy. Phép nội suy cũng là cơ sở để thực hiện các biến đổi hình học khác, ví dụ biến đổi kích thước hoặc quay ảnh, ...
- ✓ Image Processing Toolbox cung cấp ba phương pháp nội suy ảnh, bao gồm: nội suy theo các lân cận gần nhất, nội suy song tuyến tính và nội suy bicubic. Cả ba phương pháp đều thực hiện theo một nguyên tắc chung: để xác định giá trị của một pixel ảnh nội suy, ta tìm một điểm trong ảnh ban đầu tương ứng với pixel đó, sau đó giá trị của

pixel ở ảnh mới sẽ được tính bằng trung bình có trọng số của một tập các pixel nào đó ở lân cận của điểm vừa xác định, trong đó trọng số của các pixel phụ thuộc vào khoảng cách tới điểm này.

- ✓ Với phương pháp lân cận gần nhất, pixel mới sẽ được gán giá trị của pixel chứa điểm tương ứng của nó (pixel mới) trong ảnh ban đầu. Với phương pháp song tuyến tính, pixel mới sẽ được gán là trung bình có trọng số của các pixel trong một lân cận kích thước 2x2. Với phương pháp bicubic, pixel mới sẽ được gán là trung bình có trọng số của các pixel trong một lân cận kích thước 4x4.
- ✓ Phương pháp đầu tiên là phương pháp đơn giản và nhanh nhất, nhưng chất lượng không tốt bằng hai phương pháp còn lại. Số pixel được đưa vào để tính trọng số càng nhiều thì chất lượng càng tốt nhưng thời gian càng lâu. Ngoài ra, chỉ có phương pháp đầu tiên là có thể áp dụng cho mọi kiểu ảnh và kiểu dữ liệu vì nó không làm thay đổi tập giá trị của các pixel. Các phương pháp còn lại không thích hợp cho ảnh indexed, nhưng với ảnh RGB thì nên dùng các phương pháp này để bảo đảm chất lượng ảnh.
- ✓ Với ảnh RGB, phép nội suy được thực hiện một cách riêng biệt trên ba mặt phẳng màu đỏ, lam và lục.
- ✓ Với ảnh nhị phân dùng nội suy song tuyến tính hoặc bicubic, cần lưu ý đến kiểu dữ liệu, vì giá trị của pixel mới có thể nhận giá trị khác 0 và 1. Nếu ảnh gốc thuộc kiểu double thì ảnh mới sẽ là ảnh trắng đen thuộc kiểu double, nếu ảnh gốc thuộc kiểu uint8 thì ảnh mới sẽ là ảnh nhị phân kiểu uint8, trong đó các giá trị khác 0 và 1 sẽ được làm tròn về 0 hoặc 1.

Thay đổi kích thước ảnh:

- ✓ Hàm **imresize** cho phép người sử dụng thay đổi kích thước của ảnh. Ngoài kích thước ảnh mới, người sử dụng còn có thể xác định phương pháp nội suy sẽ dùng và loại bộ lọc dùng để chống aliasing.

  >> b = imresize(a,m,Method)

- ✓ Dòng lệnh trên tạo ảnh mới b có kích thước gấp m lần ảnh gốc a. Method là một chuỗi xác định phương pháp nội suy sẽ dùng: 'nearest' (lân cận gần nhất), 'bilinear' (song tuyến tính) hoặc 'bicubic'. Phương pháp mặc định là 'nearest'. Thay vì xác định tỷ số m, ta có thể xác định trực tiếp kích thước ảnh mới theo đơn vị pixel bằng cách dùng cú pháp:

  >> b = imresize(a,[mrows mcols],method)

- ✓ Trong đó mrows và mcols là số cột và số hàng của ảnh mới. Hoặc ta cũng có thể xác định cụ thể bậc của bộ lọc chống aliasing (kích thước mặc định là 11x11) hoặc cung cấp cụ thể đáp ứng xung h của bộ lọc theo các cú pháp dưới đây:

  >> b = imresize(...,method,N) % Dùng bộ lọc kích thước NxN

  >> b = imresize(...,method,h) % Dùng bộ lọc có đáp ứng xung h

Phép quay ảnh:

- ✓ Để thực hiện các phép quay ảnh, ta có thể sử dụng hàm **imrotate**. Ngoài hai thông số cơ bản là ảnh gốc và góc quay, người sử dụng cũng có thể xác định phương pháp nội suy sẽ dùng, và kích thước của ảnh mới (đủ lớn để chứa ảnh mới hay chỉ bằng kích thước ảnh cũ).Thông số mặc định là 'nearest' (lân cận gần nhất) và 'loose' (tăng kích thước nếu cần). Trong trường hợp tăng kích thước, các điểm ảnh ở ngoài phần ảnh gốc sẽ được set về 0 (màu đen). Dưới đây là cú pháp của hàm này, với Bbox là chuỗi xác định kích thước ảnh mới.

  >> b = imrotate(a,angle,Method,Bbox)

Trích xuất ảnh:

- ✓ Khi cần trích xuất một phần của ảnh gốc, ta dùng hàm **imcrop**. Khi sử dụng hàm này, người sử dụng có thể có hai lựa chọn: xác định cụ thể vị trí của phần ảnh cần trích (dưới dạng hình chữ nhật) bằng cách cung cấp các thông số vị trí khi gọi hàm hoặc sử dụng mouse để chọn phần ảnh cần trích xuất.

  Nếu chọn cách thứ nhất, ta dùng cú pháp như sau:

  >> x2 = imcrop(x,map,rect) % Ảnh indexed

  >> a2 = imcrop(a,rect) % Ảnh grayscale hoặc RGB

  trong đó *rect = [Xmin Ymin width height]*, với (Xmin,Ymin) là toạ độ góc trên bên trái của phần ảnh cần trích, width và height là chiều rộng và chiều cao của phần ảnh cần trích.

- ✓ Nếu dùng cách thứ hai, ta không cần cung cấp thông số rect, khi thực hiện hàm này, con trỏ sẽ chuyển sang dạng chữ thập, người dùng sẽ drag chuột để chọn phần ảnh

cần trích sao đó thả chuột. Hàm **imcrop** sẽ trả về phần ảnh nằm trong phạm vi xác định bởi mouse.

- ✓ Nếu không cung cấp thông số ảnh gốc, hàm **imcrop** sẽ mặc định chọn ảnh trên hệ trục tọa độ hiện hành. Ngoài ra, trong trường hợp xác định bằng mouse, người sử dụng có thể truy xuất các thông tin về vị trí và kích thước của phần ảnh đã chọn bằng cách yêu cầu thêm các output của hàm này:

  >> [A2,rect] = imcrop(A)

  >> [X2,rect] = imcrop(X,map)

Thực hiện phép thay đổi hình học tổng quát:

- ✓ Ngoài các phép biến đổi hình học cụ thể trên đây, MATLAB còn cho phép thực hiện các phép biến đổi hình học khác do người sử dụng tuỳ định bằng cách cung cấp một hàm thực hiện biến đổi hình học tổng quát, đó là hàm imtransform. Để thực hiện một phép biến đổi hình học nào đó, người sử dụng cần cung cấp ảnh cần biến đổi A và cấu trúc của phép biến đổi hình học, gọi là TFORM.

  >> B = imtransform(A,TFORM,interp)

  >> [B,XData,YData] = imtransform(...,param1,val1,param2,val2,...)

  trong đó, interp là chuỗi xác định phương pháp nội suy sẽ dùng. (Xdata, Ydata) xác định vị trí của ảnh B trong hệ trục X-Y. Ngoài ra có thể cung cấp thêm các cặp thông số (tên thông số / giá trị thông số) để xác định các thông số cụ thể của phép biến đổi. Bạn đọc có thể tìm hiểu các thông số này bằng cách gõ lệnh help imtransform từ cửa sổ lệnh của MATLAB.

- ✓ Như vậy, vấn đề quan trọng nhất khi gọi hàm này là phải xác định cấu trúc của phép biến đổi. Việc này được thực hiện bằng cách sử dụng các hàm xây dựng cấu trúc biến đổi, trong đó thông dụng nhất là hai hàm **maketform** và **cp2tform**.

  Cú pháp chung của hàm **maketform**:

  >> T = maketform(TFORM_type,...)

trong đó TFORM_type là một chuỗi xác định dạng cấu trúc biến đổi hình học, và sau đó là các thông số đi kèm tuỳ thuộc vào từng dạng cấu trúc cụ thể. Các dạng cấu trúc này được trình bày trong bảng 3.4.

✓ Hàm **cp2tform** trả về cấu trúc của phép biến đổi bàng cách suy từ các cặp điểm điều khiển trong ảnh gốc và sau khi biến đổi:

>> TFORM = cp2tform(input_points,base_points,TFORM_type,order)

input_points và base_points là các ma trận M x 2 xác định toạ độ (X,Y) của M điểm điều khiển trong ảnh biến đổi và trong ảnh gốc. TFORM_type có thể là một trong những chuỗi sau:

'linear conformal', 'affine', 'projective', 'polynomial', 'piecewise linear','lwm'. Nếu là 'polynomial' thì cần cung cấp thêm thông số order cho biết bậc của đa thức (mặc định bằng 3).

+ *Các phép biến đổi ảnh*

✓ Phép biến đổi Fourier:

Phép biến đổi Fourier biểu diễn ảnh dưới dạng tổng của các lũy thừa phức của các thành phần, biên độ, tần số và pha khác nhau của ảnh. Phép biến đổi Fourier có vai trò rất quan trọng trong các ứng dụng rộng rãi của xử lý ảnh số, bao gồm nâng cao chất lượng ảnh, phân tích, khôi phục và nén ảnh.

Do các dữ liệu trên máy tính được lưu trữ dưới dạng rời rạc, cụ thể là dữ liệu ảnh được tổ chức theo đơn vị pixel nên phép biến đổi Fourier cũng được rời rạc hóa thành biến đổi Fourier rời rạc (DFT – Discrete Fourier Transform).

Các hàm MATLAB **fft**, **fft2**, **fftn** sẽ thực hiện các phép biến đổi Fourier rời rạc 1 chiều, 2 chiều và n chiều. Các hàm **ifft**, **ifft2**, **ifftn** thực hiện các phép biến đổi DFT ngược. Với các ứng dụng xử lý ảnh, ta chỉ cần quan tâm đến các hàm **fft2** và **ifft2**.

>> F = fft2(X,Mrows,Ncols)

>> X = ifft2(F,Mrows,Ncols)

trong đó, Mrows x Ncols là kích thước của biến đổi DFT. Nếu ảnh ban đầu có kích thước nhỏ hơn thì MATLAB sẽ tự động thêm vào các zero pixel trước khi biến đổi.

Sau khi thực hiện biến đổi DFT bằng hàm **fft2**, thành phần DC của biến đổi sẽ nằm ở góc trên bên trái của ảnh. Có thể dịch chuyển thành phần này về trung tâm bằng cách dùng hàm **fftshift**.

✓ Biến đổi cosin rời rạc:

Biến đổi cosin rời rạc (DCT – Discrete cosine Transform) biểu diễn ảnh dưới dạng tổng của các cosine của các thành phần biên độ và tần số khác nhau của ảnh. Đặc điểm nổi bật của biến đổi này là: hầu hết các thông tin về ảnh chỉ tập trung trong một vài hệ số của biến đổi DCT, trong khi các hệ số còn lại chỉ chứa rất ít thông tin. Do đó, phép biến đổi này là cơ sở cho các kỹ thuật nén ảnh.

Phép biến đổi DCT thuận và nghịch được thực hiện bằng các hàm **dct2** và **idct2**. Các hàm này sử dụng giải thuật dựa theo FFT để tăng tốc độ tính toán. Hàm này thích hợp với các ảnh có kích thước lớn.

```
>> B = dct2(A,M,N)  % hoặc dct2(A,[M N]) hoặc dct2(A)
>> A = idct2(A,M,N) % hoặc dct2(B,[M N]) hoặc dct2(B)
```

✓ Biến đổi Radon:

Phép biến đổi Radon, được thực hiện bởi hàm radon trong MATLAB, biểu diễn ảnh dưới dạng các hình chiếu của nó dọc theo các hướng xác định. Hình chiếu của một hàm hai biến f(x,y) là một tập hợp các tích phân đường. Hàm radon tính các tích phân đường từ nhiều điểm nguồn dọc theo các đường dẫn song song, gọi là các tia chiếu (beam), theo một hướng xác định nào đó. Các tia chiếu này nằm cách nhau 1 pixel. Để biểu diễn toàn bộ ảnh, hàm radon sẽ lấy nhiều hình chiếu song song của ảnh từ các góc quay khác nhau bằng cách xoay các điểm nguồn quanh tâm của ảnh.

✓ Phép biến đổi fan-beam:

Phép biến đổi fan-beam cũng biểu diễn ảnh dưới dạng một tập các hình chiếu của ảnh. Một hình chiếu của một hàm hai biến f(x,y) được định nghĩa là tích phân đường của f(x,y) dọc theo các đường dẫn đồng quy tại một điểm nguồn (thay vì song song với nhau như ở biến đổi Radon). Để biểu diễn một ảnh, ta tính toán các hình chiếu tại các góc quay khác nhau khi quay điểm nguồn quanh tâm của ảnh.

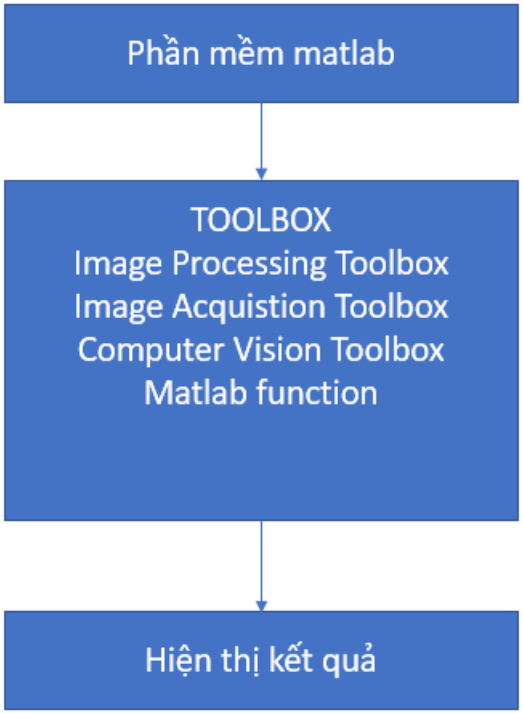
Các hàm thực hiện biến đổi fan-beam thuận và nghịch là **fanbeam** và **ifanbeam**.

>> F = fanbeam(I,D,param1,val1,param2,val2,...)

>> I = ifanbeam(F,D,param1,val1,param2,val2,...)

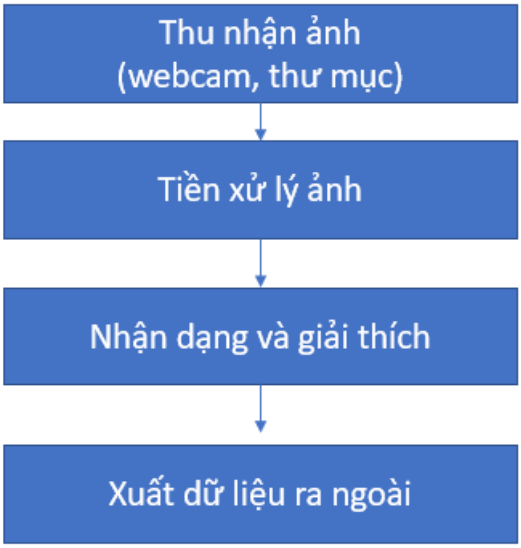
với D là khoảng cách từ điểm nguồn đến tâm của ảnh.

2.2. Mô hình tổng quát



- Theo đó, Đề tài mô phỏng nhận diện mã vạch xử dụng mô hình như hình trên.
  + Phần mềm matlab là công cụ xử lý chính
  + Sử dụng cái toolbox và xây dựng hàm để nhận diện mã vạch
  + Cuối cùng là hiện thị kết quả

2.3. Mô hình của đề tài



Mô hình của đề tài

2.4. Thu nhận hình

2.4.1 Thu nhận ảnh từ camera

Đây là bước rất quan trọng bởi nó có ảnh hưởng rất lớn đến độ chính xác trong quá trình xử lý đối với các đề tài ứng dụng xử lý ảnh. Bước này đề cập đến độ phân giải của camera, tốc độ ghi hình:

Camera được dùng trong đề tài là Image Acquisition Toolbox Support Package for OS Generic Video Interface

Gói hỗ trợ Image Acquisition Toolbox cho Giao diện Video Chung Hệ điều hành cho phép bạn thu nhận hình ảnh và video từ các thiết bị quay video.

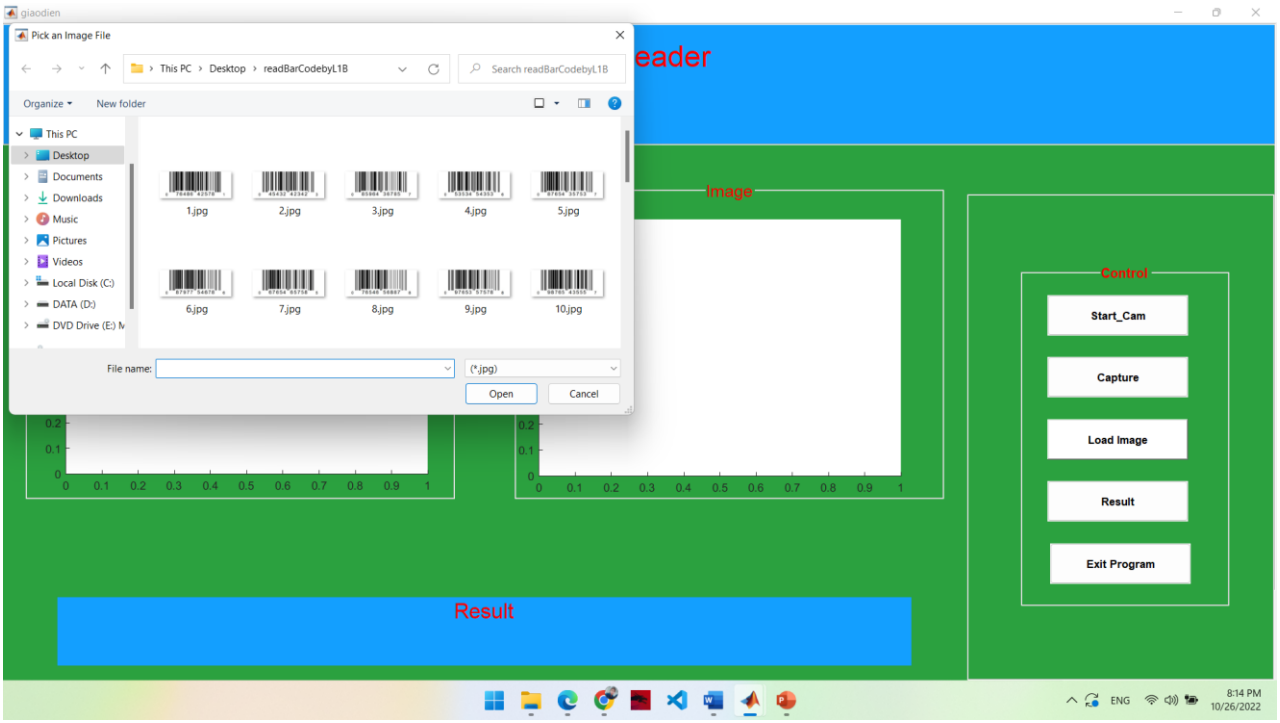
Độ phân giải có sẵn: {'1280x720' '640x480' '640x360' '320x240'}

Giao tiếp máy tính : USB

Thông số webcam được chọn trong đề tài là ‘MJPG_1280x720’

2.4.2 Thu nhận ảnh từ nguồn trong máy tính

Hình ảnh mã vạch đầu vào

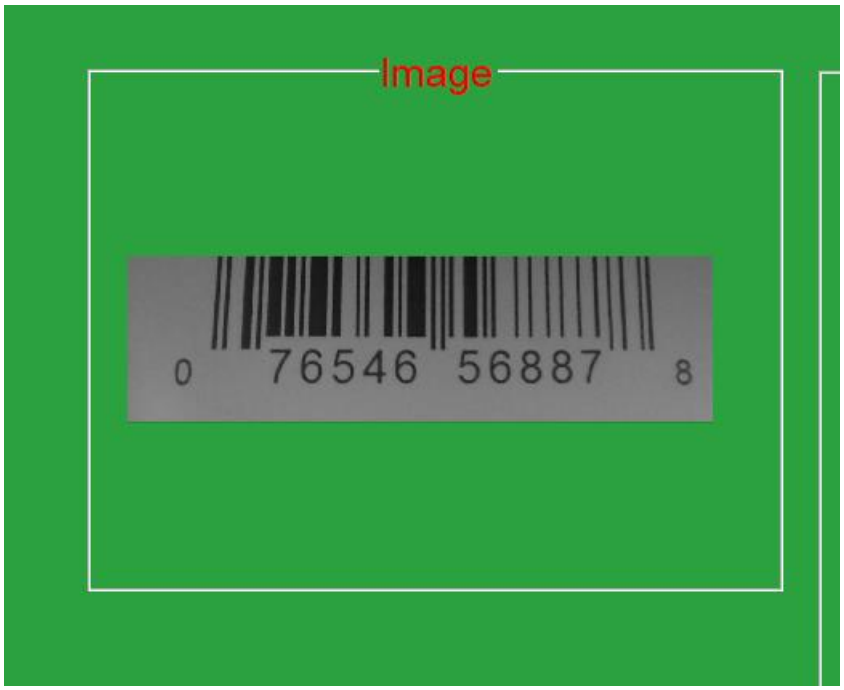


Hình ảnh thể hiện việc chọn hình ảnh

2.5. Tiền xử lý ảnh

- Ở bước này ảnh sẽ được cải thiện về độ tương phản, lọc , xám hóa… với mục đích cho chất lượng ảnh tốt hơn, trong đề tài dùng các hàm sau:

+ Lệnh rgb2gray() chuyển đổi hình ảnh RGB thành trắng đen bằng cách loại bỏ các thông tin màu sắc và độ bão hòa nhưng vẫn giữ độ sáng.



Hình ảnh sau khi thực hiện lệnh rgb2gray

+ Trước khi chuyển đổi ảnh sang ảnh nhị phân thì sẽ điều chỉnh giá trị cường độ hình ảnh, giúp làm tăng độ tương phản của hình ảnh đầu ra.

```
I = imadjust(I);
```

Ảnh sau khi được imadjust

I=imbinarize(I). chuyển đổi các hình ảnh màu xám sang một hình nhị phân. Hay tạo một ảnh nhị phân từ một ảnh cường độ, ảnh chỉ số hay ảnh RGB trên cơ sở của ngưỡng ánh sáng.

I= imbinarize (I);

Ảnh sau khi chuyển đổi thành ảnh nhị phân.

2.6. Nhận dạng và giải thích

2.7.1 Nhận diện bằng ảnh từ camera

- Dùng function: readBarcode(I) phát hiện và giải mã mã vạch 1-D hoặc 2-D trong hình ảnh đầu vào và trả về thông báo được liên kết với mã vạch đó.
- Gồm cách lệnh sau:

+ Đọc hình ảnh có chứa mã vạch vào không gian làm việc.

```
I = imread ( "barcode1D.jpg" );
```

+ Tìm kiếm hình ảnh để tìm mã vạch 1-D, trả về thông điệp, định dạng và vị trí của nó.

```
[msg, secureFormat, loc] = readBarcode (I, '1D' );
```

+ Hiển thị định dạng mã vạch được phát hiện.

```
disp ( "Định dạng mã vạch:" + secureFormat)
```

Định dạng mã vạch: EAN-13

+ Chú thích hình ảnh bằng thông báo mã vạch đã giải mã.

```
xyBegin = loc (1, :);
Imsg = insertText (I, xyBegin, msg, 'BoxOpacity' , 1, 'FontSize' , 30);
```

+ Chèn một dòng để hiển thị hàng quét.

```
imSize = size (Imsg);
Imsg = insertShape (Imsg, 'Line' , [1 xyBegin (2) imSize (2) xyBegin (2)],
'LineWidth' , 5);
```

+ Hiển thị hình ảnh.

```
imshow (Imsg)
```

2.7.2 Nhận diện bằng ảnh từ thư mục

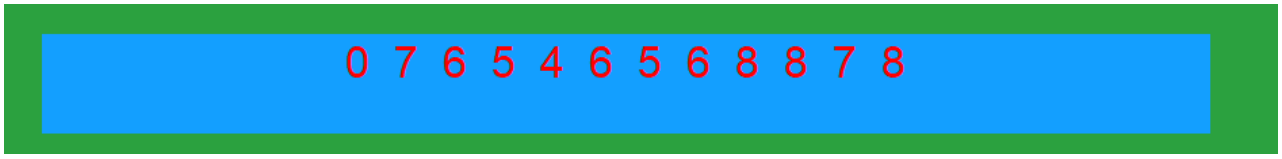
+ Bước ngày sử dụng các thuật toán để biến đổi, để xác định nhận diện mã vạch
+ Dùng hàm imcrop của Image Processing Toolbox để cắt ảnh theo vùng hình chữ nhật chứa mã vạch đã chỉ định
+ Nhận dạng mã vạch thực hiện tìm kiếm trên một số hàng được chọn của hình ảnh đầu vào, được gọi là dòng quét.

+ Các dòng quét được phân tích trên mỗi pixel và được đánh dấu theo tính năng. Sau khi tất cả các pixel được đánh dấu bằng một giá trị đặc trưng, chuỗi các mẫu sẽ được phân tích. Hàm Byhorizontal.

+ Xác định các mẫu theo trình tự và vị trí. Các ký hiệu được lấy mẫu và so sánh với sổ mã để xác định mã tương ứng. Hàm CheckT.

+ Phân tích từ trái sang phải và từ phải sang trái và chọn kết quả phù hợp hơn. Nếu tổng kiểm tra là chính xác và điểm tương ứng cao hơn một ngưỡng được chỉ định so với sổ ghi mã, mã được coi là hợp lệ và được hiển thị. Hàm eanupc

2.7. Xuất dữ liệu ra ngoài



Hình ảnh kết quả đầu ra

PHẦN 3. KẾT QUẢ MÔ PHỎNG

3.1 Tạo giao diện GUI

3.1.1 Thiết kế giao diện GUI ban đầu

Giao diện của chương trình được thiết kế như hình, bằng công cụ matlab GUI.

Với cái nút điều khiển như sau:

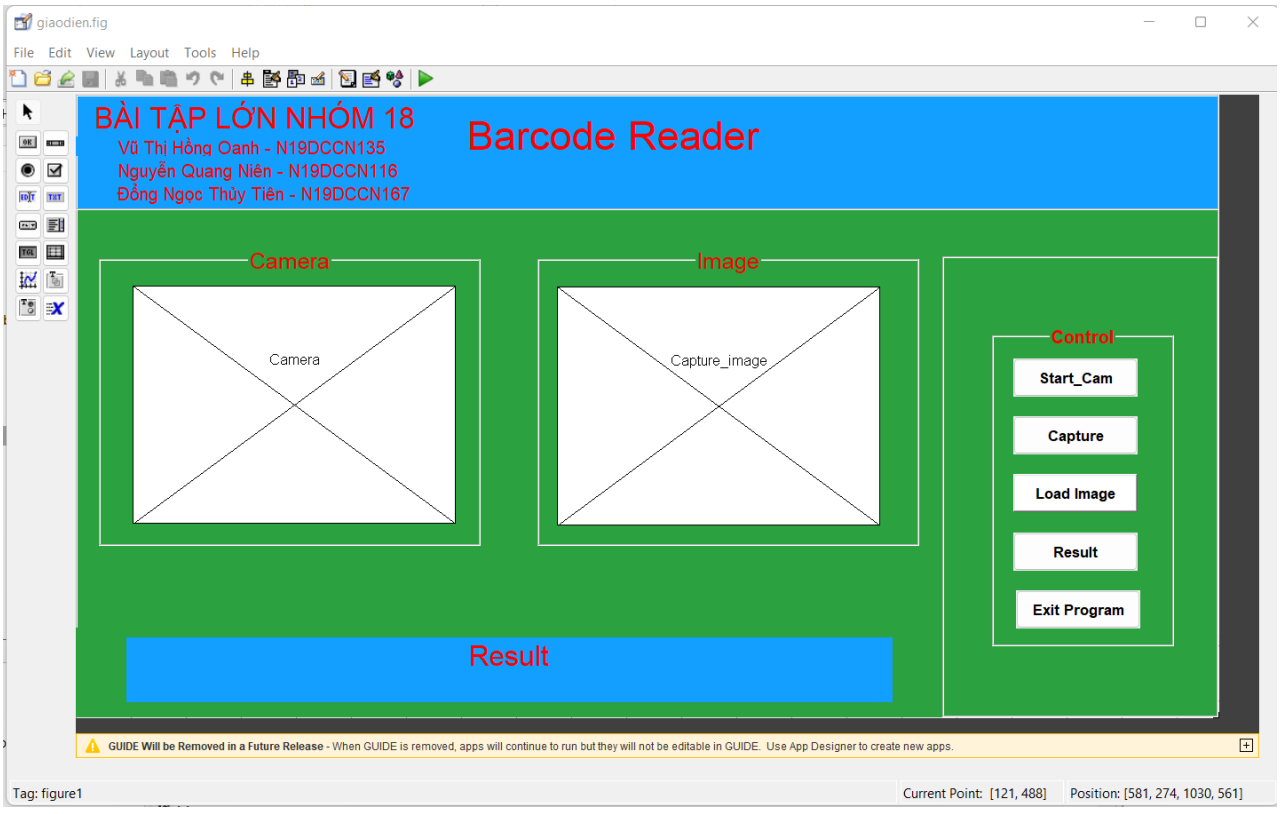
+ Nút Start_cam: camera ghi hình.

+ Nút Capture: dùng để chụp lại hình ảnh và xuất ra kết quả mã vạch đọc được.

+ Load Image: dùng để chọn ảnh từ thư mục máy tính.

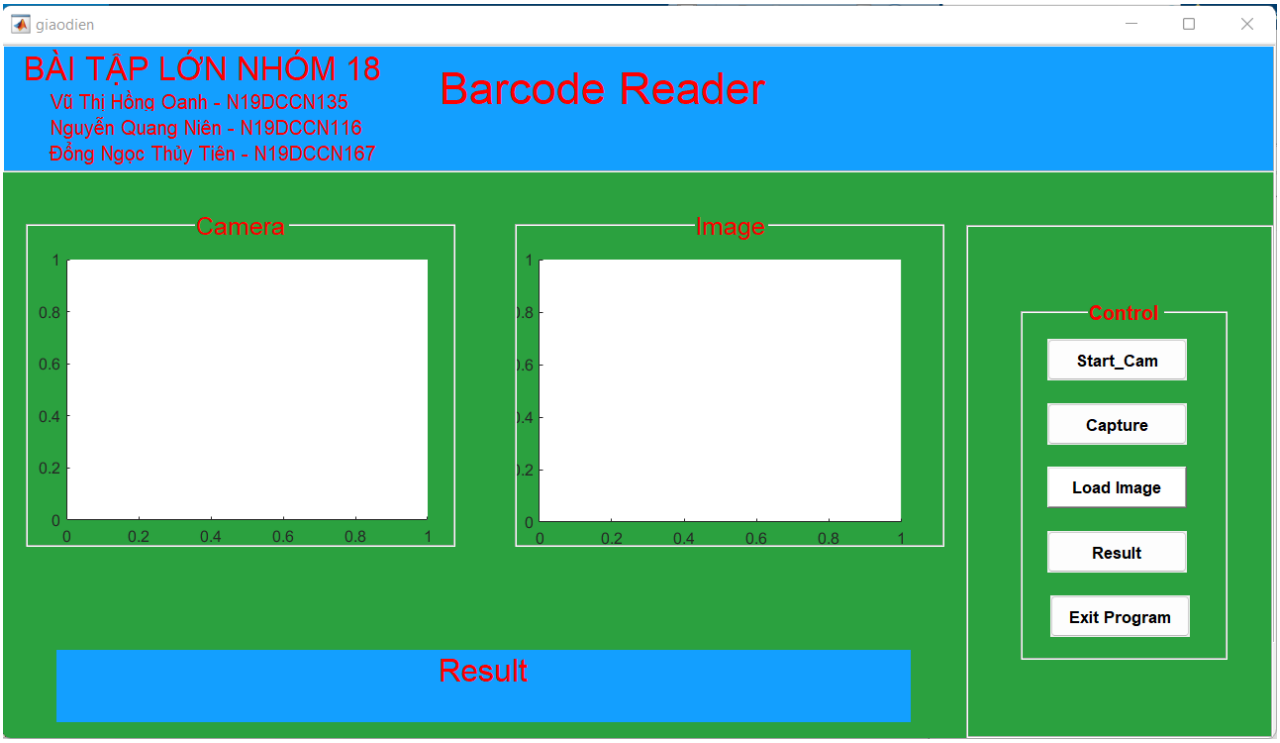
+ Result: trả kết quả mã vạch từ ảnh đọc ra từ thư mục

+ Exit Program: Thoát chương trình



Giao diện GUI ban đầu

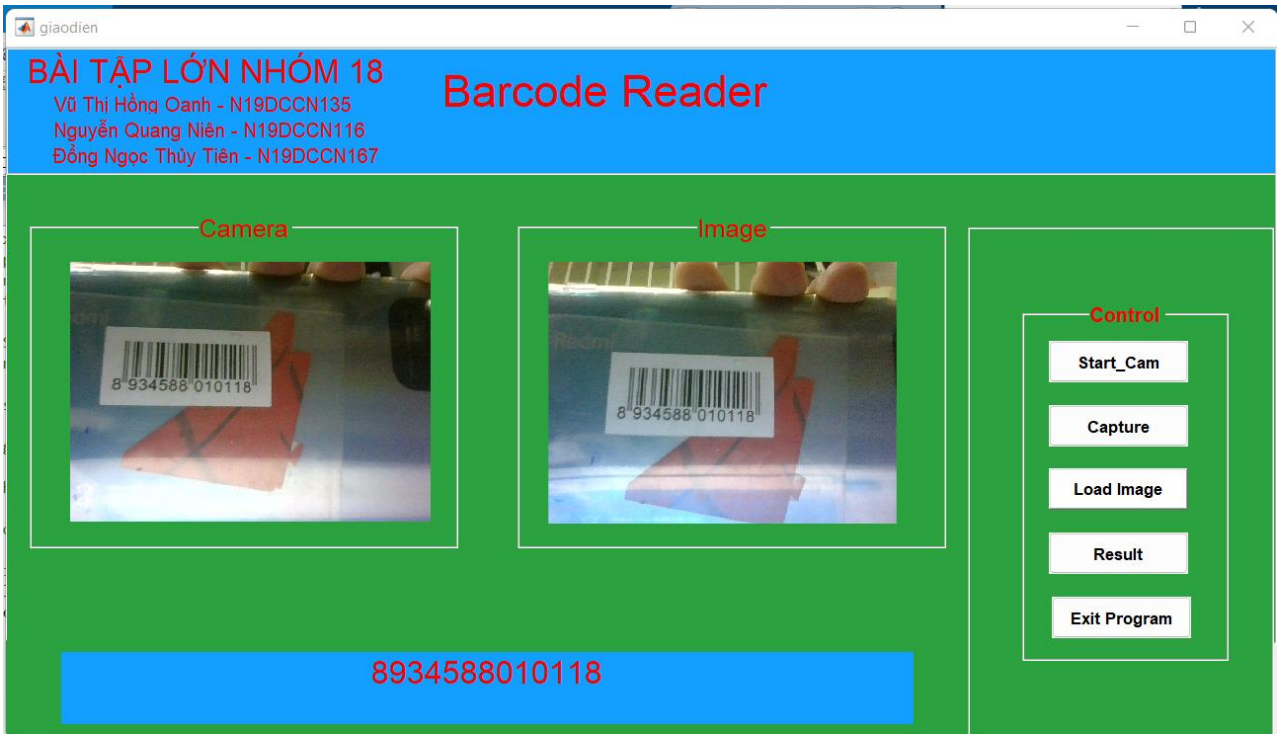
3.1.2 Giao diện GUI khi chạy chương trình



Giao diện GUI sau khi chạy chương trình

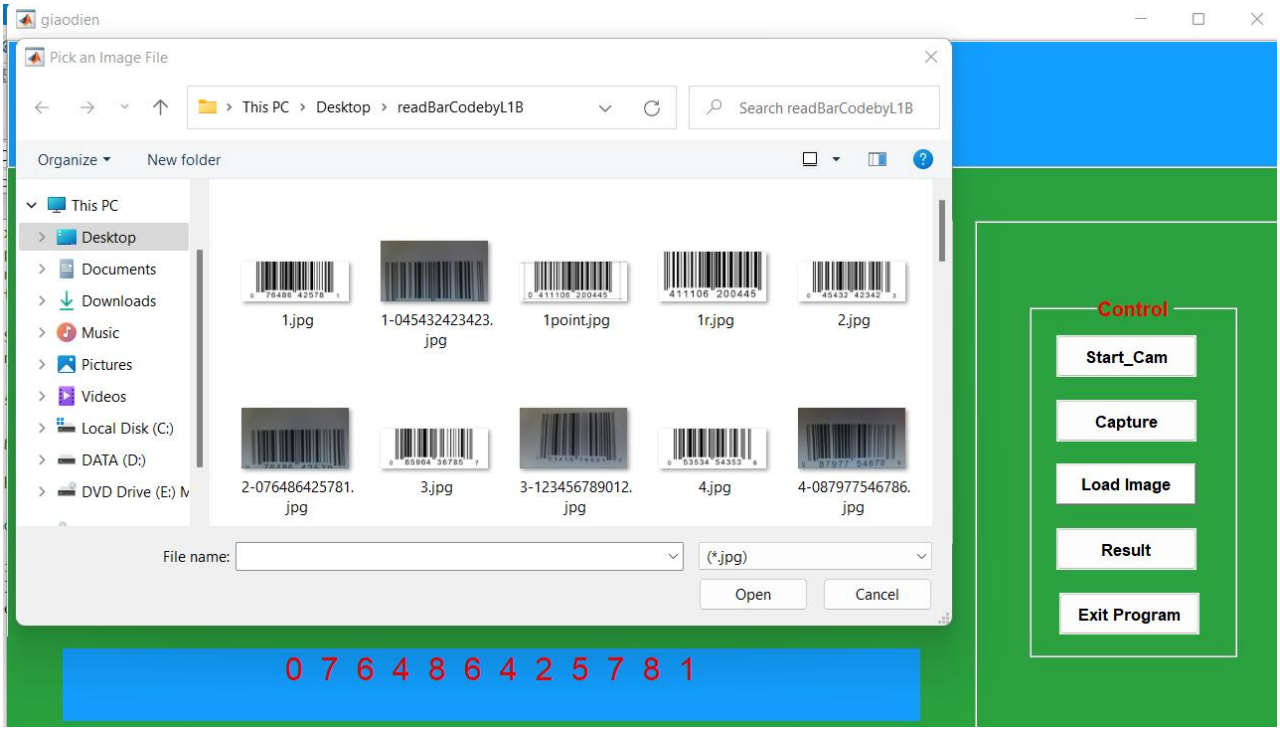
3.2 Mô phỏng kết quả

3.2.1 Mô phỏng kết quả nhận diện từ camera

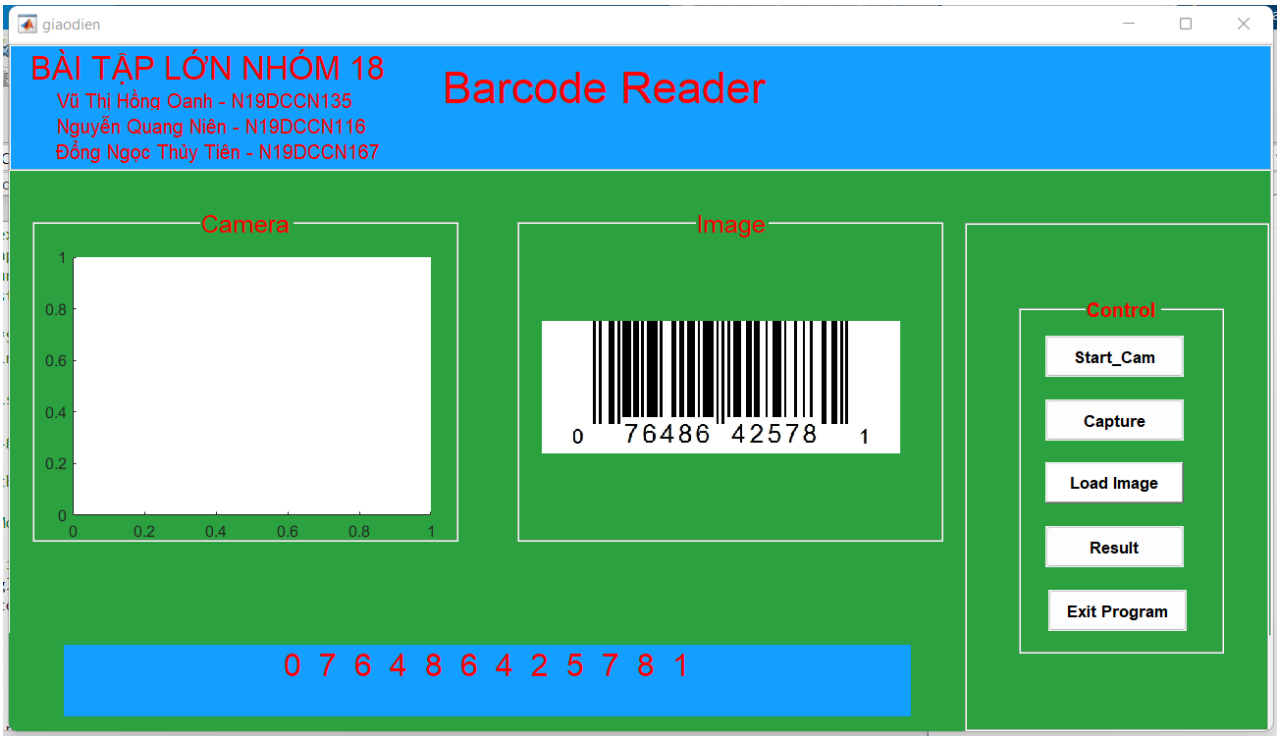


Hình ảnh mô phỏng kết quả nhận diện từ camera

3.2.2 Mô phỏng kết quả nhận diện ảnh từ thư mục máy tính



Hình ảnh mô phỏng nhận diện hình ảnh từ thư mục máy tính



PHẦN 4. KẾT LUẬN

4.1 Kết luận

Sau quá trình nghiên cứu và thực hiện đề tài đồ án "Mô Phỏng Nhận Diện Mã Vạch", đã cơ bản thực hiện được yêu cầu của đề tài là nhận diện được mã vạch từ hình hảnh cho sẵn và hình ảnh được đưa vào qua camera của máy tính. Theo kết quả mô phỏng thực tế, kết quả đọc mã vạch từ dữ liệu từ máy tính được đọc một cách chính xác, tuy nhiên dữ liệu được đọc từ camera còn bị sai lệch do chất lượng camera thu nhận hình ảnh, và bộ xử lý hình ảnh chưa được tối ưu. Kết quả trả về là dãy số hiển thị trên giao diện và trong comment window của matlab. Tuy nhiên giao diện được tạo ra còn đơn giản, chưa tạo được tính thẩm mỹ cao. việc ghi nhận hình ảnh chỉ đơn giản dùng camera máy tính dẫn đến còn hạn chế ở đối tượng sử dụng.

Ứng dụng trong quản lý bán hàng. Nhận dàng tự động thay thế ghi chép bằng tay giúp giảm nhân công, tiết kiệm thời gian, tăng hiệu suất công việc. Mã vạch được mã hóa một cách chính xác, an toàn, thông tin nhanh.

4.2 Hướng phát triển

Từ việc nghiên cứu đề tài bài tập lớn, với những mặt hạn chế của đồ án, ta nghiên cứu sâu hơn về xử lý hình ảnh sau khu thu nhận để cho ra những kết quả tối ưu hơn, nghiên cứu thay thế camera của máy tính vằng việc sử dụng các camera rời, hay các thiết bị đọc mã vạch cầm tay để dễ dàng sử dụng hơn, tiện dụng hơn, đáp ứng nhu cầu về sự thuận lợi, yêu cầu ngày càng cao của người tiêu dùng.

TÀI LIỆU THAM KHẢO

Tiếng Việt:

[1] Đề cương môn học xử lý ảnh – Học viện Công Nghệ Bưu Chính Viễn Thông Khoa CNTT1.

[2] http://baohothuonghieu.com/banquyen/tin-chi-tiet/cau-tao-ma-so-ma-vach/204.html

Tiếng Anh:

[3] R.C. Gonzalez, and R. E. Woods, “Digital Image Processing,” vol. 2, Prentice Hall, 2002.

[4] R.C. Gonzalez, R. E. Woods, “Digital Image Processing using Matlab”.

[5] https://www.mathworks.com/

[6] https://en.wikipedia.org/wiki/Barcode

Thời gian thực hiện và bảng phân công công việc của các thành viên

TT	Nội dung công việc	T1	T2	T3	T4
1	Xây dựng khung chương trình				
2	Lựa chọn hình ảnh và sản phẩm có mã vạch				
3	Xây dựng giải thuật nhận diện mã vạch				
4	Xây dựng giao diện cho chương trình(GUI)				
5	Ghép các phần lại thành chương trình hoàn chỉnh				
6	Tổng hợp và viết báo cáo cho bài tập lớn.				

Thời gian thực hiện

Phân công nội dung công việc cho từng thành viên

TT	Họ và tên	Nội dung công việc
1	Vũ Thị Hồng Oanh - N19DCCN135	Xây dựng giải thuật nhận diện mã vạch
		Xây dựng khung chương trình

		Ghép các phần lại thành chương trình hoàn chỉnh.
2	Nguyễn Quang Niên - N19DCCN116	Xây dựng giải thuật nhận diện mã vạch
		Tổng hợp và viết báo cáo cho bài tập lớn
3	Đổng Ngọc Thủy Tiên - N19DCCN167	Xây dựng giao diện cho chương trình(GUI)
		Lựa chọn hình ảnh và sản phẩm có mã vạch

Bảng phân công công việc

PHỤ LỤC NGHIÊN CỨU

SOURCE CODE

File giao diện.m

```
 function varargout = giaodien(varargin)
% GIAODIEN M-file for giaodien.fig
%      GIAODIEN, by itself, creates a new GIAODIEN or raises the existing
%      singleton*.
%
%      H = GIAODIEN returns the handle to a new GIAODIEN or the handle to
%      the existing singleton*.
%
%      GIAODIEN('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GIAODIEN.M with the given input arguments.
%
%      GIAODIEN('Property','Value',...) creates a new GIAODIEN or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before pcc1_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to giaodien_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Copyright 2002-2003 The MathWorks, Inc.

% Edit the above text to modify the response to help giaodien
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @giaodien_OpeningFcn, ...
                   'gui_OutputFcn',  @giaodien_OutputFcn, ...
```

```
                'gui_LayoutFcn',  [] , ...
                'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before giaodien is made visible.
function giaodien_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to giaodien (see VARARGIN)
% Choose default command line output for giaodien
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes giaodien wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = giaodien_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on button press in start_cam.
function start_cam_Callback(hObject, eventdata, handles)
global vid;
    cla(handles.Camera,'reset');
```

```
    guidata(hObject, handles); %updates the handles
    imaqreset;
    set(gcf,'CurrentAxes',handles.Camera);
    set(gcf,'DoubleBuffer','on');
    imaqhwinfo;
    imaqhwinfo('winvideo',1);
    vid=videoinput('winvideo',1,'MJPG_640x480');
    vid.ReturnedColorSpace = 'rgb';
    vidRes = get(vid, 'VideoResolution');
    nBands = get(vid, 'NumberOfBands');
    hImage = image( zeros(vidRes(2), vidRes(1), nBands) );
    preview(vid, hImage);
% hObject    handle to start_cam (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in exitprogram.
function exitprogram_Callback(hObject, eventdata, handles)
delete(handles.figure1);
% hObject    handle to exitprogram (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in result.
function result_Callback(hObject, eventdata, handles)
S = handles.S;      %image
NRows=size(S,1);
NColoms=size(S,2);
S=imcrop(S,[0 NRows/2 NColoms NRows/1.2]);
depth=10;
horizantalRule=depth+1;
I=rgb2gray(S);
I = imadjust(I);
I=imbinarize(I);
hang=size(I,1);
cot=size(I,2);
%By Horizontal Lines
for i=1:(horizantalRule-1)
```

```
    c=BYhorizontal(horizantalRule,hang,cot,I);              %function# 1

    td=Region(c);                                           %function# 2

    s=Collect(c,td);                                        %function# 3

    [k,g]=CheckT(s);                                        %function# 4

    rec=s(k);
    q=s./rec;
    p=round(q);

    if (td-g-k)==57 & p(k)==1 & p(k+1)==1 & p(k+2)==1

        result=eanupc(p,k)


    if ischar(result)
    set(handles.result_barcode,'string',result);
      else
    result=num2str(result);
    set(handles.result_barcode,'string',result);
      end


       break;
    end
end
clear
% hObject    handle to result (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in capture.
function capture_Callback(hObject, eventdata, handles)
clc;
global vid;
```

```
    S=getsnapshot(vid);
    axes(handles.Capture_image);
    imshow(S);
    %Image Storage
    imwrite(S,'imagename2.jpg');
    %%%%
    handles.S = S;
    guidata(hObject, handles);
    I =
imread("C:\Users\HongOanh\OneDrive\Desktop\readBarCodebyL1B\imagename2.jpg");
    [msg,detectedFormat,loc] = readBarcode(I,'1D');
    disp("Decoded barcode message: " + msg)
    disp("Barcode format: " + detectedFormat)
    set(handles.result_barcode,'string',msg);
   % hObject    handle to capture (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in loadimage.
function loadimage_Callback(hObject, eventdata, handles)
[filename, pathname] = uigetfile({'*.jpg';'*.bmp';'*.jpg';'*.gif'}, 'Pick an Image File');
S = imread([pathname,filename]);
axes(handles.Capture_image);
imshow(S);
handles.S = S;
guidata(hObject, handles);
% hObject    handle to loadimage (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes during object creation, after setting all properties.
function result_barcode_CreateFcn(hObject, eventdata, handles)
% hObject    handle to result_barcode (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% --- Executes during object creation, after setting all properties.
function Capture_image_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Capture_image (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: place code in OpeningFcn to populate Capture_image
% Hint: place code in OpeningFcn to populate Capture_image
% --- Executes during object creation, after setting all properties.
function axes3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: place code in OpeningFcn to populate axes3
% --- Executes during object creation, after setting all properties.
function axes4_CreateFcn(hObject, eventdata, handles)
tato=imread('logo.jpg');
imshow(tato);
% hObject    handle to axes4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: place code in OpeningFcn to populate axes4
% --- Executes during object creation, after setting all properties.
function logo_CreateFcn(hObject, eventdata, handles)
imshow(imread('logo.gif'))
% hObject    handle to logo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: place code in OpeningFcn to populate logo
```

File Byhorizontal.m

```
function c =BYhorizontal(horizantalRule,NumofRows,NumofColoms,I)

   j=1;
   b=[1:NumofColoms];
```

```
for x=1:NumofColoms
   if (I(round(NumofRows/horizantalRule),x)==0)
      b(j)=1;
      j=j+1;
   elseif (I(round(NumofRows/horizantalRule),x)==1)
      b(j)=0;
      j=j+1;
   end
end

c=b;
```