**Bộ Giáo Dục Và Đào Tạo**

**Trường Đại Học Ngoại Ngữ - Tin Học Thành Phố Hồ Chí Minh**

**Khoa Công Nghệ Thông Tin**



**MÔN HỌC : THỊ GIÁC MÁY TÍNH**

**ĐỀ TÀI : PHÂN VÙNG ẢNH DỰA TRÊN NGƯỠNG**

**Giáo Viên Hướng Dẫn** : PGS.TS Nguyễn Thanh Bình

**Thành Viên :**

1. Trần Ngọc Thảo Ngân – MSSV: 21DH114460
2. Bùi Tuấn Đạt - MSSV: 21DH113218
3. Nguyễn Hoàng Gia Huy - MSSV: 21DH110660
4. Cao Thảo Vân - MSSV: 21DH113129

*Tp. Hồ chí minh, Ngày 17 Tháng 11 năm 2023*

**Nhận xét của giảng viên**

**Lời cảm ơn**

Để hoàn thành tốt đề tài này chúng em xin cảm ơn tới quí thầy cô trường HUFLIT đã tạo điều kiện cho chúng em nghiên cứu và hoàn thành đề tài. Tiếp đến, chúng em xin chân trọng cảm ơn giáo viên bộ môn Thị giác máy tính thầy Nguyễn Thanh Bình đã tận tình giúp đỡ, chỉ bảo hướng dẫn chúng em trong suốt quá trình thực hiện đề tài. Vì thời gian và năng lực còn có hạn chế nên không thể tránh khỏi những sai sót trong khi thực hiện đề tài nghiên cứu của mình. Rất mong được sự góp ý bổ sung của thầy để đề tài của chúng em ngày càng hoàn thiện hơn.

Chúng em xin chân thành cảm ơn !

# Mục lục

[Mục lục 4](#_heading=h.gjdgxs)

[1.](#_heading=h.30j0zll) CHƯƠNG 1 : GIỚI THIỆU 7

[**1.1.**](#_heading=h.1fob9te) **Giới thiệu đề tài** 7

[**1.2.**](#_heading=h.3znysh7) **Mục tiêu và nội dung của đề tài** 7

[**1.3.**](#_heading=h.2et92p0) **Giới hạn đề tài** 8

[**1.4.**](#_heading=h.tyjcwt) **Cấu trúc báo cáo** 8

[2.](#_heading=h.3dy6vkm) CHƯƠNG 2 : CƠ SỞ LÝ THUYẾT LIÊN QUAN 9

[3.](#_heading=h.2s8eyo1) CHƯƠNG 3 : PHƯƠNG PHÁP OTSU 12

[**3.1.**](#_heading=h.17dp8vu) **Yêu cầu bài toán** 12

[**3.2.**](#_heading=h.3rdcrjn) **Giải thuật thực hiện** 12

[**3.3.**](#_heading=h.35nkun2) **Phương pháp đánh giá** 13

[4.](#_heading=h.1y810tw) CHƯƠNG 4 : HIỆN THỰC 19

[**4.1.**](#_heading=h.4i7ojhp) **Yêu cầu hệ thống và tập dữ liệu thực nghiệm** 19

[**4.2.**](#_heading=h.2xcytpi) **Hiện thực kết quả** 19

[5.](#_heading=h.147n2zr) CHƯƠNG 5 : KẾT LUẬN 30

[Tài liệu tham khảo 32](#_heading=h.3o7alnk)

[Phụ lục code demo 33](#_heading=h.23ckvvd)

**Danh mục hình**

[Hình 1. Ví dụ phân vùng ảnh 9](#_heading=h.1t3h5sf)

[Hình 2. Biểu đồ histogram của f(x,y) 10](#_heading=h.4d34og8)

[Hình 3. Giải thuật cho toàn bài toán ứng dụng 12](#_heading=h.26in1rg)

[Hình 4. Giải thuật chi tiết cho phân vùng ảnh bằng thuật toán tìm ngưỡng Otsu 13](#_heading=h.lnxbz9)

[Hình 5. Ảnh sau khi thực hiện phân vùng và tính thời gian thực thi (thuật toán được xây dựng lại) 16](#_heading=h.1ksv4uv)

[Hình 6. Thời gian thực thi của phân vùng ảnh bằng thuật toán tìm ngưỡng Otsu (opencv) 16](#_heading=h.44sinio)

[Hình 7. Ảnh phân vùng bằng thuật toán tìm ngưỡng Otsu (opencv) 17](#_heading=h.2jxsxqh)

[Hình 8. Kết quả của phép đo độ lỗi giữa 2 ảnh 17](#_heading=h.z337ya)

[Hình 9. Ảnh sau khi được hệ thống chấm điểm 18](#_heading=h.3j2qqm3)

[Hình 10. Ảnh đầu vào test\_grader.png 20](#_heading=h.1ci93xb)

[Hình 11. Ảnh sau khi phát hiện biên cạnh 21](#_heading=h.3whwml4)

[Hình 12. Ảnh bài kiểm tra được tìm thấy với đường viền bao quanh 22](#_heading=h.2bn6wsx)

[Hình 13. Ảnh nhìn từ trên xuống với ảnh gốc](about:blank) 23

[Hình 14. Ảnh nhìn từ trên xuống đã chuyển sang ảnh xám 23](#_heading=h.qsh70q)

[Hình 15. Ảnh sau khi thực hiện phân vùng 26](#_heading=h.3as4poj)

[Hình 16. Ảnh sau khi tìm được các ô tròn 27](#_heading=h.1pxezwc)

[Hình 17. Ảnh các dòng câu hỏi được viền bằng các màu khác nhau 28](#_heading=h.49x2ik5)

[Hình 18. Ảnh kết quả sau khi chấm điểm 29](#_heading=h.2p2csry)

**Bảng phân công công việc**

| **Thành viên**  **(Tên + MSSV)** | **Nhiệm vụ** |
| --- | --- |
| Trần Ngọc Thảo Ngân - 21DH114460 | * Viết code xây dựng hệ thống * Chương 3: Phương pháp Otsu   + 3.2 Giải thuật thực hiện   + 3.3 Phương pháp đánh giá * Chương 4: Hiện thực kết quả   + 4.2 Kết quả thực nghiệm * Tổng hợp báo cáo |
| Cao Thảo Vân - 21DH113129 | * Chương 1: Giới thiệu   + 1.1 Giới thiệu đề tài   + 1.2 Mục tiêu và nội dung đề tài   + 1.3 Giới hạn đề tài   + 1.2 Cấu trúc báo cáo |
| Nguyễn Hoàng Gia Huy - 21DH110660 | * Chương 2: Cơ sở lý thuyết * Chương 3: Phương pháp Otsu   + 3.1 Yêu cầu bài toán * Chương 4: Hiện thực kết quả   + 4.1 Yêu cầu hệ thống |
| Bùi Tuấn Đạt - 21DH113218 | * Chương 5: Kết luận   + 5.1 Kết quả đạt được   + 5.2 Ưu và nhược điểm   + 5.3 Hướng mở rộng tương lai |

# CHƯƠNG 1 : GIỚI THIỆU

## **Giới thiệu đề tài**

Trong lĩnh vực thị giác máy tính, việc xử lý và phân tích ảnh là một chủ đề quan trọng, đặc biệt là khi chúng ta muốn tự động hóa quá trình nhận biết và hiểu thông tin từ hình ảnh. Phương pháp phân vùng ảnh dựa trên ngưỡng là một trong những kỹ thuật cơ bản nhưng hiệu quả, được sử dụng rộng rãi để tách biệt các đối tượng hoặc khu vực quan trọng trong ảnh.

Bài toán phân vùng ảnh dựa trên ngưỡng đóng vai trò quan trọng trong nhiều ứng dụng thực tế, và một trong những ứng dụng đặc biệt quan trọng là hệ thống chấm điểm tự động cho bài kiểm tra trắc nghiệm. Trong quá trình chấm điểm bài kiểm tra trắc nghiệm, việc tự động phân vùng ảnh giúp nhận diện và đánh giá đáp án của học sinh một cách nhanh chóng và chính xác, giảm thiểu sai sót và tăng tính hiệu quả của quá trình chấm điểm.

Trong đề tài này, chúng ta sẽ tập trung vào việc áp dụng phương pháp phân vùng ảnh bằng thuật toán tìm ngưỡng Otsu để xây dựng một hệ thống chấm điểm tự động cho bài kiểm tra trắc nghiệm. Bằng cách phân vùng ảnh, hệ thống có thể tự động nhận dạng và đếm số lượng câu hỏi và các lựa chọn trên mỗi bài kiểm tra.

## **Mục tiêu và nội dung của đề tài**

Mục tiêu chính của đề tài là sử dụng kỹ thuật phân vùng ảnh dựa trên ngưỡng để xây dựng một hệ thống chấm điểm tự động cho bài kiểm tra trắc nghiệm. Hệ thống này sẽ có các mục tiêu cụ thể sau:

* Phát triển thuật toán phân vùng ảnh chính xác và linh hoạt, có khả năng xử lý nhiều loại bài kiểm tra và điều kiện ánh sáng khác nhau
* Xác định vị trí và biên của các ô trả lời trên tờ bài kiểm tra.
* Tính toán và ghi điểm tự động dựa trên đáp án mẫu được định nghĩa trước.

Nội dung của đề tài sẽ bao gồm các phần chính sau:

* Nghiên cứu về phương pháp phân ngưỡng hình ảnh: Tìm hiểu về ngưỡng hình ảnh và cách nó được sử dụng để phân vùng ảnh. Chọn ra phương pháp phân vùng ảnh dựa trên ngưỡng phù hợp nhất để xây dựng hệ thống chấm điểm bài kiểm tra trắc nghiệm. Ở đây chúng ta lựa chọn phương pháp phân vùng ảnh bằng thuật toán tìm ngưỡng Otsu để xây dựng hệ thống.
* Xây dựng hệ thống chấm điểm tự động: Phát triển một hệ thống tự động để chấm điểm các bài kiểm tra đa lựa chọn. Quá trình này sẽ bao gồm việc tiền xử lý hình ảnh, phân vùng hình ảnh để xác định các vùng quan trọng (câu hỏi và các phương án trả lời), và áp dụng các thuật toán khác để chấm điểm và đưa ra kết quả cuối cùng.
* Đánh giá hiệu suất: Tiến hành các thử nghiệm và đánh giá hiệu suất của thuật toán tìm ngưỡng Otsu đã được xây dựng lại không sử dụng hàm cv2.threshold của thư viện opencv trong việc tìm ngưỡng để phân vùng ảnh, và đồng thời cũng đánh giá kết quả của hệ thống chấm điểm tự động

## **Giới hạn đề tài**

Đề tài tập trung vào việc nghiên cứu và phát triển hệ thống chấm điểm tự động cho bài kiểm tra trắc nghiệm sử dụng kỹ thuật phân vùng ảnh bằng thuật toán tìm ngưỡng Otsu. Và sử dụng phép đo độ lỗi MSE để đánh giá độ chính xác, tính thời gian thực thi của phương pháp phân vùng ảnh bằng thuật toán tìm ngưỡng Otsu để đánh giá hiệu suất của thuật toán sau khi được xây dựng lại.

Đề tài chủ yếu tập trung vào việc sử dụng thuật toán tìm ngưỡng Otsu để thực hiện quá trình phân vùng ảnh và ứng dụng vào bài toán thực tế. Các phương pháp phân vùng khác không được xem xét hay tích hợp trong nghiên cứu.

Hệ thống được xây dựng để chấm điểm cho loại hình ảnh bài kiểm tra trắc nghiệm mà câu hỏi có 5 phương án trả lời, phương án trả lời được vẽ bằng hình tròn và không áp dụng cho các loại hình ảnh bài kiểm tra khác. Nếu muốn áp dụng được hệ thống cho các loại hình ảnh bài kiểm tra khác thì cần phải thay đổi, hiệu chỉnh các thông số như số phương án trả lời của mỗi câu hỏi, hình dạng của phương án trả lời, đáp chính xác của hình kiểm tra đang xét,…

## **Cấu trúc báo cáo**

Chương 1: Giới thiệu về đề tài và tầm quan trọng của đề tài áp dụng trong cuộc sống, mục tiêu cụ thể và nội dung của đề tài, giới hạn đề tài.

Chương 2: Chương này cho ta biết rõ hơn về các thuật toán, các công thức của từng thuật toán, cách áp dụng và công dụng của các thuật toán đó.

Chương 3: Chương này tụi em phân tích bài toán ứng dụng, đưa ra các yêu cầu của bài toán, các phương pháp giải quyết và phương pháp đánh giá.

Chương 4: Hiện thực bài toán, cho biết các yêu cầu phần cứng phần mềm và cho ra kết quả của bài toán ứng dụng.

Chương 5: Kết luận đồ án đã đạt hay chưa đạt được những gì, ưu và nhược điểm của phương pháp đề xuất và cuối cùng là hướng mở rộng tương lai.

# CHƯƠNG 2 : CƠ SỞ LÝ THUYẾT LIÊN QUAN

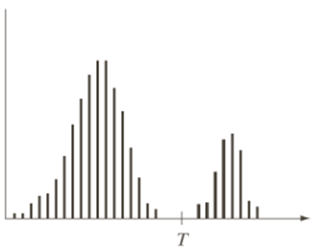
* 1. **Phân vùng ảnh**
* Phân vùng ảnh là quá trình chia bức ảnh thành các vùng đồng nhất hoặc các đối tượng có cùng đặc điểm theo các tiêu chí định trước.
* Các tiêu chí có thể là: cường độ, lược đồ xám, giá trị trung bình (mean), phương sai (variance),…
  + Ví dụ: Để giảm nhiễu của nền vào quá trình nhận dạng người ta đánh dấu vùng nền bằng màu đen

A collage of a person walking on a sidewalk

Description automatically generated

*Hình 1. Ví dụ phân vùng ảnh*

* 1. **Phân vùng ảnh dựa vào kỹ thuật cắt ngưỡng**
* Khái niệm:
* Ảnh đa cấp xám (ảnh xám): là ảnh mà các pixel có giá trị xám từ 0 … 255 (Thông thường)
* Ảnh màu (RGB): mỗi pixel có 03 thành phần đại diện cho 3 màu (Red, Green, Blue)
* Trong đó: giá trị Red: từ 0 … 255; Giá trị Green: từ 0 … 255; Giá trị Blue: từ 0 … 255
* Cắt ngưỡng là kỹ thuật chuyển giá trị mức xám của ảnh (pixel) về một giá trị xám chung nào đó.Ví dụ cắt ngưỡng nhị phân là chuyển ảnh về 2 giá trị: hoặc 0 hoặc 1
* Phân loại:
* Cắt ngưỡng toàn cục (giá trị ngưỡng đơn lẻ và không đổi)
* Cắt ngưỡng đa ngưỡng (nhiều giá trị ngưỡng)
* Cắt ngưỡng thích nghi (giá trị ngưỡng thay đổi và phụ thuộc giá trị mức xám của pixel cắt ngưỡng và các pixel lân cận)
  1. **Phân vùng ảnh dựa vào kỹ thuật cắt ngưỡng toàn cục**
* **Cắt ngưỡng toàn cục (đơn ngưỡng):** cắt ngưỡng dựa vào 1 giá trị ngưỡng T cho toàn bộ ảnh



*Hình 2. Biểu đồ histogram của f(x,y)*

=> Phân chia ảnh vào 2 vùng: Vùng có mức xám = 1 và vùng có mức xám = 0

* Như vậy:
  + Phân vùng ảnh bằng kỹ thuật cắt ngưỡng thì phải xác định ngưỡng T
  + Ngưỡng T được áp dụng cho toàn bộ ảnh nên gọi là Ngưỡng toàn cục
* Thuật toán tìm ngưỡng: là những thuật toán tìm ngưỡng để áp dụng phân đoạn ảnh dựa trên kỹ thuật cắt ngưỡng toàn cục
  1. **Thuật toán tìm ngưỡng Otsu**

Thuật toán tìm ngưỡng Otsu là một phương pháp tự động xác định ngưỡng tối ưu để phân vùng ảnh thành hai lớp: một lớp chứa các điểm ảnh có giá trị nhỏ hơn ngưỡng và một lớp chứa các điểm ảnh có giá trị lớn hơn ngưỡng.

Nguyên lý hoạt động:

* B1: Gọi ni là số pixel có mức xám I 🡪 tính xác suất pixel có mức xám I
* B2: Chọn giá trị ngưỡng ban đầu k (bất kỳ lớn hơn hoặc bằng 0)
* B3: Tạo ra 2 nhóm điểm ảnh:

Nhóm A chứ các pixel với mức xám <= k ([0…k])

Nhóm B chứa các pixel với mức xám > k ([k+1 …L-1])

* B4: Tính xác suất tích lũy đến mức xám k của nhóm A là P1(k) và xác suất tích lũy đến mức xám L-1 của nhóm B là P2(k)

* B5: Tính giá trị trung bình xác suất trong nhóm A là m1(k) và nhóm B là m2(k)
* B6: Tính giá trị trung bình toàn cục của mức xám ảnh:
* B7: Tính giá trị phương sai giữa 2 nhóm A và B:

+

* B8: Giá trị ngưỡng được tính:

# CHƯƠNG 3 : PHƯƠNG PHÁP OTSU

## **Yêu cầu bài toán**

Bài toán ứng dụng: Hệ thống chấm điểm tự động cho bài kiểm tra trắc nghiệm

Yêu cầu:

* Phát hiện được vị trí của bài kiểm tra trong ảnh
* Áp dụng được thuật toán Otsu để phân biệt giữa vùng đối tượng và vùng nền
* Trích xuất các ô tròn phương án trả lời
* Xác định được câu trả lời đã đánh dấu
* Tính điểm

## **Giải thuật thực hiện**

* Các bước giải thuật cho toàn bài toán ứng dụng

A diagram of a work flow

Description automatically generated

*Hình 3. Giải thuật cho toàn bài toán ứng dụng*

* Giải thuật chi tiết cho phân vùng ảnh bằng thuật toán tìm ngưỡng Otsu

A diagram of a flowchart

Description automatically generated

*Hình 4. Giải thuật chi tiết cho phân vùng ảnh bằng thuật toán tìm ngưỡng Otsu*

## **Phương pháp đánh giá**

Đánh giá bằng phép đo độ lỗi MSE (mean\_squared\_error) và thời gian thực thi

* Với việc phân vùng ảnh bằng thuật toán tìm ngưỡng Otsu đã được xây dựng lại như code bên dưới thì thời gian thực thi tính được là khoảng 1 phút 30 giây (tùy vào tốc độ mạng). Thời gian thực thi tương đối là lâu hơn nhiều so với thời gian thực thi của phương pháp phân vùng ảnh bằng thuật toán tìm ngưỡng Otsu sử dụng hàm cv2.threshold của thư viện opencv (khoảnh 0.005 giây). 🡪 Hiệu suất thực hiện phân vùng ảnh của thuật toán tìm ngưỡng Otsu được xây dựng lại chưa tốt lắm.
* Tuy nhiên khi sử dụng phép đo độ lỗi MSE để so sánh giữa hai phương pháp phân vùng ảnh bằng thuật toán tìm ngưỡng Otsu có sử dụng thư viện và không sử dụng thư viện (được xây dựng lại) thì ta có kết quà là 0.0. Điều này cho thấy hình ảnh sau khi phân vùng bằng thuật toán tìm ngưỡng Otsu (được xây dựng lại) có độ chính xác cao và giống với ảnh phân vùng sử dụng thư viện
* Các code thực hiện như phụ lục đính kèm bên dưới

def otsu(img):

    phuong\_sai\_t = 0

    M,N= img.shape

    mG = np.mean(img)   # Tính mG giá trị trung bình mức xám của ảnh theo công thức 6

    for nguong in range(256):

        Tong\_gt\_xam\_A = 1  #Khởi tạo biến lưu tổng giá trị mức xám của nhóm A, B

        Tong\_gt\_xam\_B = 1

        Tong\_pixel\_A = 1    #Khởi tạo biến lưu tổng số pixel ở nhóm A, B

        Tong\_pixel\_B = 1

        for i in range(M):  #Duyệt qua giá trị xám của mỗi pixel của hình ảnh gốc

            for j in range(N):

                if (img[i,j] >= nguong):  #Nếu pixel có giá trị màu xám > = nguong (nhóm A)

                    Tong\_pixel\_A = Tong\_pixel\_A + 1

                    Tong\_gt\_xam\_A = Tong\_gt\_xam\_A + img[i,j]

                else:     #Nếu pixel có giá trị xám < nguong (nhóm B)

                    Tong\_pixel\_B = Tong\_pixel\_B + 1

                    Tong\_gt\_xam\_B = Tong\_gt\_xam\_B + img[i,j]

        P1 = Tong\_pixel\_A/(M\*N) # Tính P1(k), P2(k) theo công thức (xác suất tích lũy)

        P2 = Tong\_pixel\_B/(M\*N)

        m1 = Tong\_gt\_xam\_A/Tong\_pixel\_A # Tính m1(k), m2(k) theo công thức (giá trị trung bình xác suất)

        m2 = Tong\_gt\_xam\_B/Tong\_pixel\_B

        phuong\_sai = P1\*((m1-mG)\*\*2)+P2\*((m2-mG)\*\*2) # Tính phương sai theo công thức

        if (phuong\_sai > phuong\_sai\_t): # xác định phương sai tối đa theo công thức

            phuong\_sai\_t = phuong\_sai

            nguong\_toi\_uu = nguong  # Để có được ngưỡng tối ưu của phương sai tối đa

    print("Ngưỡng tìm được", nguong\_toi\_uu)

    return nguong\_toi\_uu

def phan\_vung\_bang\_cat\_nguong\_otsu(img,nguong):

    m, n = img.shape

    img\_phan\_vung\_cat\_nguong = np.zeros([m, n])

    for i in range(m):

        for j in range(n):

            if (img[i,j] < nguong):

                img\_phan\_vung\_cat\_nguong[i,j] = 255

            else:

                img\_phan\_vung\_cat\_nguong[i,j] = 0

    return img\_phan\_vung\_cat\_nguong

if \_\_name\_\_ == "\_\_main\_\_":

    start\_time = time.time()

    nguong= otsu(warped)

    img\_phan\_vung = phan\_vung\_bang\_cat\_nguong\_otsu(warped,nguong)

    fig2 = plt.figure(figsize=(16, 9))

    (ax1, ax2), (ax3, ax4) = fig2.subplots(2, 2)

    # Hiển thị ảnh gốc

    ax1.imshow(warped, cmap='gray', alpha = 0.9)

    ax1.set\_title('Ảnh gốc')

    ax1.axis('off')

    # Hiển thị histogram ảnh gốc

    ax2.hist(warped.flatten(), bins=256)

    ax2.set\_title('Hitogram ảnh gốc')

    # Hiển thị ảnh phân vùng

    ax3.imshow(img\_phan\_vung, cmap='gray')

    ax3.set\_title('Ảnh phân vùng dựa vào tìm ngưỡng Otsu')

    ax3.axis('off')

    # Hiển thị histogram ảnh phân vùng

    ax4.hist(img\_phan\_vung.flatten(), bins=256)

    ax4.set\_title('Hitogram ảnh phân vùng')

    plt.show()

    end\_time = time.time()

    execution\_time = end\_time - start\_time

    print(f"Thời gian thực thi của hàm otsu là {execution\_time} giây")

A graph and diagram of a graph

Description automatically generated with medium confidence

*Hình 5. Ảnh sau khi thực hiện phân vùng và tính thời gian thực thi (thuật toán được xây dựng lại)*

start\_time = time.time()

thresh = cv2.threshold(warped, 0, 255,cv2.THRESH\_BINARY\_INV | cv2.THRESH\_OTSU)[1]

end\_time = time.time()

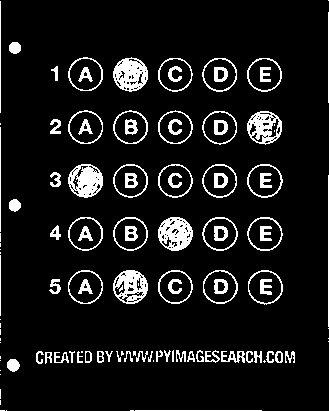
execution\_time = end\_time - start\_time

print(f"Thời gian thực thi của hàm otsu (opencv) là {execution\_time} giây")

cv2\_imshow(thresh)



*Hình 6. Thời gian thực thi của phân vùng ảnh bằng thuật toán tìm ngưỡng Otsu (opencv)*



*Hình 7. Ảnh phân vùng bằng thuật toán tìm ngưỡng Otsu (opencv)*

MSE = mean\_squared\_error(thresh, img\_phan\_vung)

MSE



*Hình 8. Kết quả của phép đo độ lỗi giữa 2 ảnh*

* Khi so sánh kết quả sau khi chấm điểm với đáp án được lưu vào lúc ban đầu thì hệ thống đã chấm điểm chính xác, đã khoanh vùng được câu trả lời đúng và cả câu trả lời sai của người làm bài kiểm tra.

ANSWER\_KEY = {

    0: 1,

    1: 4,

    2: 0,

    3: 3,

    4: 1}

* Tương ứng với đáp áp đúng cho bài kiểm tra là:

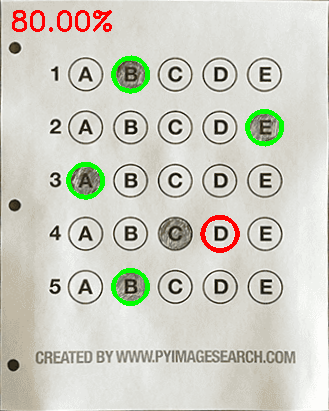
Câu 1: B

Câu 2: E

Câu 3: A

Câu 4: D

Câu 5: B



*Hình 9. Ảnh sau khi được hệ thống chấm điểm*

# CHƯƠNG 4 : HIỆN THỰC

## **Yêu cầu hệ thống và tập dữ liệu thực nghiệm**

* Lập trình bằng ngôn ngữ Python
* Chạy trên nền tảng Google Colab
* Yêu cầu phần cứng:
* Bộ Nhớ (RAM): Ít nhất 4GB RAM để đảm bảo khả năng xử lý ảnh và dữ liệu một cách hiệu quả.
* Bộ Nhớ Trống trên Ổ Cứng: Ít nhất 10GB để lưu trữ chương trình và dữ liệu liên quan.
* Tập dữ liệu thực nghiệm: file chứa các hình ảnh được chụp từ bài kiểm tra của người làm bài (file Images đính kèm trong source code)

## **Hiện thực kết quả**

Các code thực hiện như phụ lục đính kèm bên dưới

* Đầu tiên là import các thư viện cần thiết

from imutils.perspective import four\_point\_transform

from imutils import contours

import numpy as np

import argparse

import imutils

import cv2

from matplotlib import pyplot as plt

from google.colab.patches import cv2\_imshow

import time

from skimage.metrics import mean\_squared\_error

* Sử dụng ANSWER\_KEY để lưu đáp án cho từng câu hỏi trong bài trắc nghiệm

ANSWER\_KEY = {

    0: 1,

    1: 4,

    2: 0,

    3: 3,

    4: 1}

* Đáp áp đúng cho bài kiểm tra là:

Câu 1: B

Câu 2: E

Câu 3: A

Câu 4: D

Câu 5: B

* Đọc ảnh đầu vào

image = cv2.imread("/content/drive/MyDrive/Colab Notebooks/test\_grader.png")

cv2\_imshow(image)

A paper with a test chart

Description automatically generated with medium confidence

*Hình 10. Ảnh đầu vào test\_grader.png*

* Xử lý ảnh đầu vào : chuyển đổi ảnh gốc sang ảnh xám, làm mở ảnh bằng bộ lọc Gausian để giảm nhiễu và làm cho các chi tiết trên ảnh trở nên mượt mà hơn. Sau đó sử dụng phương pháp Canny để phát hiện biên cạnh trong ảnh.

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

blurred = cv2.GaussianBlur(gray, (5, 5), 0)

edged = cv2.Canny(blurred, 75, 200)

cv2\_imshow(edged)

A screen shot of a computer screen

Description automatically generated

*Hình 11. Ảnh sau khi phát hiện biên cạnh*

* Sau khi xác định được biên cạnh, chúng ta sử dụng cv2.findContours tìm các đường bao quanh (contour) trong ảnh

cnts = cv2.findContours(edged.copy(), cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

cnts = imutils.grab\_contours(cnts)

docCnt = None

* Chúng ta thực hiện sắp xếp các contours theo diện tích của chúng (từ lớn đến nhỏ) sau khi đảm bảo rằng ít nhất một contour đã được tìm thấy. Điều này ngụ ý rằng các contours lớn sẽ được đặt ở đầu danh sách, trong khi các contours nhỏ sẽ xuất hiện ở phía sau trong danh sách.

if len(cnts)>0:

    cnts= sorted(cnts, key=cv2.contourArea, reverse=True)

* Tiếp theo, chúng ta kiểm tra số đỉnh trên contour.

Chúng ta lặp qua từng đường contour đã được sắp xếp. Đối với mỗi contour, chúng ta tiến hành xấp xỉ, tức là giảm số điểm trên contour, làm cho nó trở nên đơn giản hơn.

Chúng ta kiểm tra xem đường biên đã xấp xỉ có bốn điểm hay không, và nếu có thì chúng ta đã tìm thấy bài kiểm tra.

    for c in cnts:

        peri= cv2.arcLength(c, True)

        approx= cv2.approxPolyDP(c, 0.02\*peri, True)

        if(len(approx) == 4):

            docCnt= approx

            break

contourImage = image.copy()

cv2.drawContours(contourImage, [docCnt], -1, (0, 0, 255), 2)

cv2\_imshow(contourImage)

A paper with a number on it

Description automatically generated

*Hình 12. Ảnh bài kiểm tra được tìm thấy với đường viền bao quanh*

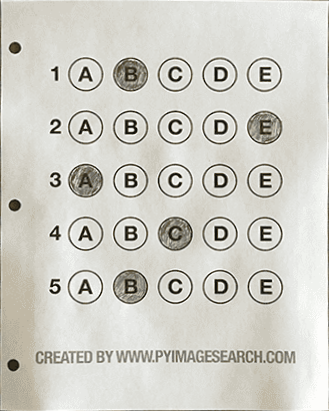
* Sau đó chúng ta áp dụng phép biến đổi tọa độ theo góc bốn điểm để có được một cái nhìn từ trên xuống đối với tờ giấy bài kiểm tra.

paper = four\_point\_transform(image, docCnt.reshape(4, 2))

warped = four\_point\_transform(gray, docCnt.reshape(4, 2))

cv2\_imshow(paper)

cv2\_imshow(warped)



A sheet of paper with circles and letters

Description automatically generated

*Hình 14. Ảnh nhìn từ trên xuống đã chuyển sang ảnh xám*

* Chúng ta sử dụng thuật toán Otsu để tự động xác định giá trị ngưỡng phù hợp để phân biệt giữa vùng nền và vùng đối tượng trong ảnh. Sau khi tìm được ngưỡng dựa trên thuật toán Otsu, dựa trên ngưỡng tìm được chúng ta sẽ tiến hành phân vùng ảnh, trong đó vùng nền được biểu diễn bằng màu đen (0) và các vùng đối tượng được biểu diễn bằng màu trắng (255)

def otsu(img):

    phuong\_sai\_t = 0

    M,N= img.shape

    mG = np.mean(img)   # Tính mG giá trị trung bình mức xám của ảnh theo công thức 6

    for nguong in range(256):

        Tong\_gt\_xam\_A = 1  #Khởi tạo biến lưu tổng giá trị mức xám của nhóm A, B

        Tong\_gt\_xam\_B = 1

        Tong\_pixel\_A = 1    #Khởi tạo biến lưu tổng số pixel ở nhóm A, B

        Tong\_pixel\_B = 1

        for i in range(M):  #Duyệt qua giá trị xám của mỗi pixel của hình ảnh gốc

            for j in range(N):

                if (img[i,j] >= nguong):  #Nếu pixel có giá trị màu xám > = nguong (nhóm A)

                    Tong\_pixel\_A = Tong\_pixel\_A + 1

                    Tong\_gt\_xam\_A = Tong\_gt\_xam\_A + img[i,j]

                else:     #Nếu pixel có giá trị xám < nguong (nhóm B)

                    Tong\_pixel\_B = Tong\_pixel\_B + 1

                    Tong\_gt\_xam\_B = Tong\_gt\_xam\_B + img[i,j]

        P1 = Tong\_pixel\_A/(M\*N) # Tính P1(k), P2(k) theo công thức (xác suất tích lũy)

        P2 = Tong\_pixel\_B/(M\*N)

        m1 = Tong\_gt\_xam\_A/Tong\_pixel\_A # Tính m1(k), m2(k) theo công thức (giá trị trung bình xác suất)

        m2 = Tong\_gt\_xam\_B/Tong\_pixel\_B

        phuong\_sai = P1\*((m1-mG)\*\*2)+P2\*((m2-mG)\*\*2) # Tính phương sai theo công thức

        if (phuong\_sai > phuong\_sai\_t): # xác định phương sai tối đa theo công thức

            phuong\_sai\_t = phuong\_sai

            nguong\_toi\_uu = nguong  # Để có được ngưỡng tối ưu của phương sai tối đa

    print("Ngưỡng tìm được", nguong\_toi\_uu)

    return nguong\_toi\_uu

def phan\_vung\_bang\_cat\_nguong\_otsu(img,nguong):

    m, n = img.shape

    img\_phan\_vung\_cat\_nguong = np.zeros([m, n])

    for i in range(m):

        for j in range(n):

            if (img[i,j] < nguong):

                img\_phan\_vung\_cat\_nguong[i,j] = 255

            else:

                img\_phan\_vung\_cat\_nguong[i,j] = 0

    return img\_phan\_vung\_cat\_nguong

if \_\_name\_\_ == "\_\_main\_\_":

    start\_time = time.time()

    nguong= otsu(warped)

    img\_phan\_vung = phan\_vung\_bang\_cat\_nguong\_otsu(warped,nguong)

    fig2 = plt.figure(figsize=(16, 9))

    (ax1, ax2), (ax3, ax4) = fig2.subplots(2, 2)

    # Hiển thị ảnh gốc

    ax1.imshow(warped, cmap='gray', alpha = 0.9)

    ax1.set\_title('Ảnh gốc')

    ax1.axis('off')

    # Hiển thị histogram ảnh gốc

    ax2.hist(warped.flatten(), bins=256)

    ax2.set\_title('Hitogram ảnh gốc')

    # Hiển thị ảnh phân vùng

    ax3.imshow(img\_phan\_vung, cmap='gray')

    ax3.set\_title('Ảnh phân vùng dựa vào tìm ngưỡng Otsu')

    ax3.axis('off')

    # Hiển thị histogram ảnh phân vùng

    ax4.hist(img\_phan\_vung.flatten(), bins=256)

    ax4.set\_title('Hitogram ảnh phân vùng')

    plt.show()

    end\_time = time.time()

    execution\_time = end\_time - start\_time

    print(f"Thời gian thực thi của hàm otsu là {execution\_time} giây")

A graph and diagram of a graph

Description automatically generated with medium confidence

*Hình 15. Ảnh sau khi thực hiện phân vùng*

* Sau khi phân vùng ảnh thì chúng ta tiến hành tìm các contour trên ảnh đã được phân vùng. Tiếp theo, khởi tạo mảng questionCnts để lưu các contour tương ứng với các ô tròn trên bài kiểm tra.
* Để xác định khu vực nào trên ảnh là các ô tròn, chúng ta lặp qua từng contour. Đối với mỗi contour, chúng ta tính bounding box và tỉ lệ giữa chiều rộng và chiều cao để xác định đúng ô tròn cần tìm. Để 1 khu vực của contour được coi là ô tròn thì khu vực đó cần đủ rộng và cao (ít nhất là 20 pixel ở cả hai chiều); có tỉ lệ giữa chiều rộng và chiều cao gần bằng 1.

img\_phan\_vung = img\_phan\_vung.astype(np.uint8)

cnts = cv2.findContours(img\_phan\_vung.copy(), cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

cnts = imutils.grab\_contours(cnts)

questionCnts = []

for c in cnts:

(x, y, w, h) = cv2.boundingRect(c)

ar = w / float(h)

if w >= 20 and h >= 20 and ar >= 0.9 and ar <= 1.1:

questionCnts.append(c)

questionsContourImage = paper.copy()

cv2.drawContours(questionsContourImage, questionCnts, -1, (0, 0, 255), 3)

cv2\_imshow(questionsContourImage)

A sheet of paper with red circles

Description automatically generated

*Hình 16. Ảnh sau khi tìm được các ô tròn*

* Chúng ta sẽ đến phần chấm điểm cho bài kiếm tra
* Đầu tiên chúng ta sắp xếp questionCnts từ trên xuống dưới, sau đó khởi tạo số lượng câu trả lời đúng . Sau đó chúng ta sẽ bắt đầu lặp qua các câu hỏi. Vì mỗi câu hỏi có 5 phương án trả lời, nên chúng ta sẽ áp dụng cắt mảng Numpy và sắp xếp các contour của câu hỏi hiện tại từ trái sang phải .

import random

questionCnts = contours.sort\_contours(questionCnts, method="top-to-bottom")[0]

correct = 0

questionsContourImage = paper.copy()

for (q, i) in enumerate(np.arange(0, len(questionCnts), 5)):

  cnts = contours.sort\_contours(questionCnts[i:i + 5])[0]

  cv2.drawContours(questionsContourImage, cnts, -1, (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255)), 2)

  bubbled = None

A paper with circles and letters

Description automatically generated

*Hình 17. Ảnh các dòng câu hỏi được viền bằng các màu khác nhau*

* Tiếp theo là xác định ô tròn nào đã được điền. Xây dựng một mask cho ô tròn hiện tại, sau đó đếm số lượng pixel non-zero trong ô tròn hiện tại. Trong dòng câu hỏi đang xét thì ô tròn nào có số lượng pixel non-zero tối đa thì đó là ô tròn được người làm bài đánh dấu là câu trả lời.

  for (j, c) in enumerate(cnts):

    mask = np.zeros(img\_phan\_vung.shape, dtype="uint8")

    cv2.drawContours(mask, [c], -1, 255, -1)

    mask = cv2.bitwise\_and(thresh, thresh, mask=mask)

    total = cv2.countNonZero(mask)

    if bubbled is None or total > bubbled[0]:

      bubbled = (total, j)

* So sánh câu trả lời của người làm bài với đáp án đúng. Nếu người làm bài đúng, chúng ta sẽ khoanh bằng màu xanh. Tuy nhiên, nếu người làm bài làm sai, chúng ta sẽ khoanh câu trả lời đúng bằng màu đỏ

  color = (0, 0, 255)

  k = ANSWER\_KEY[q]

  if k == bubbled[1]:

    color = (0, 255, 0)

    correct += 1

  cv2.drawContours(paper, [cnts[k]], -1, color, 3)

cv2\_imshow(questionsContourImage)

* Cuối cùng chúng ta sẽ tính điểm cho bài kiểm tra và hiển thị điểm lên màn hình. [1]

score = (correct / 5.0) \* 100

print(f"Score: {score:.2f}%")

cv2.putText(paper, f"{score:.2f}%", (10, 30),

            cv2.FONT\_HERSHEY\_SIMPLEX, 0.9, (0, 0, 255), 2)

print("Ảnh gốc")

cv2\_imshow(image)

print("Ảnh sau khi chấm điểm")

cv2\_imshow(paper)

Score: 80.00%

A test sheet with green circles and a red circle

Description automatically generated

*Hình 18. Ảnh kết quả sau khi chấm điểm*

# CHƯƠNG 5 : KẾT LUẬN

* 1. **Kết quả đạt được**

Thông qua báo cáo này chúng em đã đạt được thành công trong việc phát triển một hệ thống chấm điểm tự động cho bài kiểm tra trắc nghiệm bằng việc áp dụng với kỹ thuật phân vùng ảnh dựa trên thuật toán tìm ngưỡng Otsu.

Đã thiết lập và hiện thực quy trình chấm điểm từ việc xác định vùng bài kiểm tra, phân vùng ảnh đến việc đánh giá câu trả lời và tính điểm.

Báo cáo đã đánh giá hiệu suất của kỹ thuật phân vùng ảnh dựa trên thuật toán tìm ngưỡng Otsu sau khi được xây dựng lại, tính được độ chính xác giữa ảnh phân vùng có sử dụng thư viện và không sử dụng thư viện, so sánh được độ chính xác của điểm sau khi chấm và đáp án bài kiểm tra được lưu vào lúc đầu.

* 1. **Ưu và nhược điểm**

**Ưu điểm:**

* Tích hợp thuật toán Otsu: Việc tích hợp thuật toán tìm ngưỡng Otsu đã giúp tự động xác định ngưỡng phân vùng, làm cho hệ thống trở nên linh hoạt và dễ sử dụng.
* Chấm điểm tự động: Hệ thống có khả năng chấm điểm tự động, giảm độ phức tạp và tăng hiệu suất so với việc thực hiện thủ công.
* Đơn giản hóa quy trình chấm điểm: Hệ thống giúp đơn giản hóa quy trình chấm điểm bài kiểm tra trắc nghiệm, giảm thiểu công sức và thời gian so với việc chấm điểm thủ công.
* Độ chính xác cao: độ chính xác cao trong việc sử dụng kỹ thuật phân vùng ảnh bằng thuật toán tìm ngưỡng Otsu để phân vùng ảnh, và chấm điểm cho cho bài kiểm tra.

**Nhược điểm:**

* Giới hạn ảnh bài kiểm tra cụ thể: Hệ thống phù hợp cho loại ảnh bài kiểm tra trắc nghiệm cụ thể và không tự động thích ứng với tất cả các định dạng khác nhau.
* Hiệu suất của phương pháp phân vùng ảnh bằng thuật toán tìm ngưỡng Otsu chưa cao, thời gian thực thi còn khá lâu sao với việc phân vùng mà sử dụng hàm cv2.threshold.
  1. **Hướng mở rộng tương lai**
* Mở rộng áp dụng cho nhiều loại bài kiểm tra: Nghiên cứu và phát triển để hệ thống có thể tự động áp dụng cho nhiều loại bài kiểm tra khác nhau mà không cần sự can thiệp thủ công.
* Tăng hiệu suất cho phương pháp phân vùng ảnh: Nghiên cứu và tích hợp thêm các thư viện khác trong tính toán để phương pháp phân vùng ảnh bằng thuật toán tìm ngưỡng Otsu sau khi được xậy dựng lại có hiệu suất cao hơn, thời gian thực thi thấp hơn.
* Tích hợp trí tuệ nhân tạo: Nghiên cứu và tích hợp các kỹ thuật trí tuệ nhân tạo để cải thiện khả năng nhận diện và chấm điểm, đặc biệt là đối với các trường hợp khó nhận diện.

# Tài liệu tham khảo

* Image segmentation(Stefano Ferrari): <https://homes.di.unimi.it/ferrari/ImgProc2011_12/EI2011_12_16_segmentation_double.pdf> (truy cập lần cuối 17/11/2023)
* Thresholding-Based Image Segmentation: <https://www.geeksforgeeks.org/thresholding-based-image-segmentation/> (truy cập lần cuối 17/11/2023)
* A Survey on Threshold Based Segmentation Technique in Image Processing: <https://www.researchgate.net/profile/Singaraju-Jyothi/publication/309209325_A_Survey_on_Threshold_Based_Segmentation_Technique_in_Image_Processing/links/5805bb6f08aee314f68e2879/A-Survey-on-Threshold-Based-Segmentation-Technique-in-Image-Processing.pdf> (truy cập lần cuối 17/11/2023)
* Bubble sheet multiple choice scanner and test grader using OMR, Python, and OpenCV: <https://pyimagesearch.com/2016/10/03/bubble-sheet-multiple-choice-scanner-and-test-grader-using-omr-python-and-opencv/> (truy cập lần cuối 17/11/2023)

# Phụ lục code demo

<https://colab.research.google.com/drive/1Iq8l5f80V5ziXO1GyDITbQKkghmA52yt?usp=sharing>