TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN GIỮA KÌ

NHẬP MÔN XỬ LÝ ẢNH SỐ

TÌM HIỂU VỀ ADAPTIVE THRESHOLDING

Người hướng dẫn: **GV PHẠM VĂN HUY**

Người thực hiện: **LÊ HOÀNG LONG – 518H0035**

TRẦN VĂN AN – 518H0127

Lớp : **18H50205**

Khoá : **22**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2020

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN GIỮA KÌ MÔN NHẬP MÔN XỬ LÝ ẢNH SỐ

TÌM HIỂU VỀ ADAPTIVE THRESHOLDING

Người hướng dẫn: **GV PHẠM VĂN HUY**
Người thực hiện: **LÊ HOÀNG LONG**
TRẦN VĂN AN
Lớp **: 18H50205**
Khoá **: 22**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2020

LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn tới giảng viên bộ môn thầy Phạm Văn Huy đã dạy dỗ, giảng dạy nhiệt tình và dành thời gian quý báu giúp chúng em hiểu và làm bài dễ hơn, trong quá trình học có một vài tuần gặp khó khăn do ảnh hưởng dịch Covid-19 phải học online nhưng thầy vẫn tạo điều kiện giúp đỡ sinh viên gặp khó khăn trong quá trình làm bài. Do chưa có nhiều kinh nghiệm làm để tài, trong bài tiểu luận chắc chắn sẽ không tránh khỏi những thiếu sót. Rất mong nhận được sự nhận xét, ý kiến đóng góp, phê bình từ phía thầy và các bạn. Em chúc thầy luôn nhiều sức khỏe để tiếp tục đóng góp cho sự nghiệp giáo dục của nước nhà.

ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi / chúng tôi và được sự hướng dẫn của GV Phạm Văn Huy;. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày tháng năm

Tác giả

(ký tên và ghi rõ họ tên)

Lê Hoàng Long

Trần Văn An

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

Phần đánh giá của GV chấm bài

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

TÓM TẮT

Bài báo cáo nhằm giải thích tại sao lại sử dụng Adaptive Thresholding và nó là gì, được ứng dụng như nào.

Contents

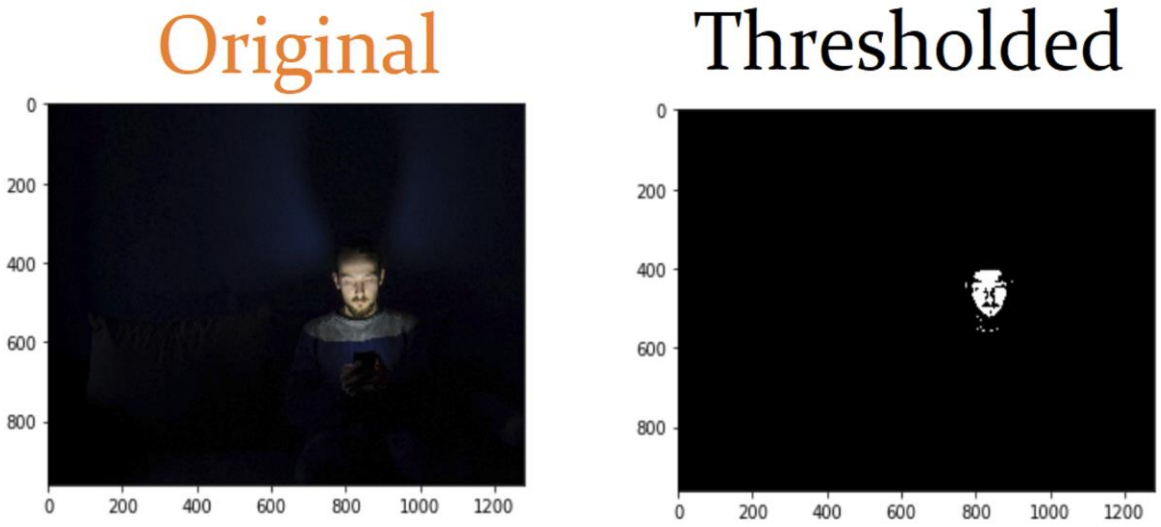
CHƯƠNG 1

THRESHOLDING LÀ GÌ VÀ TẠI SAO LẠI DÙNG ADAPTIVE THRESHOLDING?

1.1 Thresholding là gì?

Thresholding là một kỹ thuật trong OpenCV, là việc gán các giá trị pixel liên quan đến giá trị ngưỡng đã cung cấp. Trong quá trịnh thresholding, mỗi giá trị pixel được so sánh với giá trị ngưỡng. Nếu giá trị pixel nhỏ hơn ngưỡng thì nó sẽ đặt thành 0 và ngược là nó được đặt là bằng giá trị max (thường là 255). Thresholding là một kỹ thuật phân đoạn, được sử dụng để xét một đối tượng khỏi các nền của nó trong bức ảnh. Ngưỡng là một giá trị có hai vùng ở hai bên tức là dưới ngưỡng hoặc trên ngưỡng.

Trong Thị Giác Máy Tính, kỹ thuật xác định ngưỡng được thực hiện trên ảnh có thang độ xám. Do đó hình nahr ban đầu phải được chuyển đổi trong không gian màu và thang độ xám.



1.1.1 Ngưỡng cơ bản:

Kỹ thuật Ngưỡng cơ bản hay còn gọi là Ngưỡng nhị phân. Đối với mọi pixel, cùng một giá trị ngưỡng được áp dụng. Nếu giá trị pixel nhỏ hơn ngưỡng thì đặt thành 0 ngược lại đặt thành giá trị tối đa.

Có tất cả 5 ngưỡng cơ bản:

1. cv2.THRESH_BINARY
2. cv2.THRESH_BINARY_INV
3. cv2.THRESH_TRUNC
4. cv2.THRESH_TOZERO
5. cv2.THRESH_TOZERO_INV

Binary

$$\texttt{dst}(x,y) = \begin{cases} \texttt{maxval} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$$

Inverted Binary

$$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{maxval} & \text{otherwise} \end{cases}$$

Truncated

$$\texttt{dst}(x,y) = \begin{cases} \texttt{threshold} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$$

To Zero

$$\texttt{dst}(x,y) = \begin{cases} \texttt{src}(x,y) & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$$

To Zero Inverted

$$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$$

1.1.2 Code minh họa:

Đây là cú pháp trong OpenCV

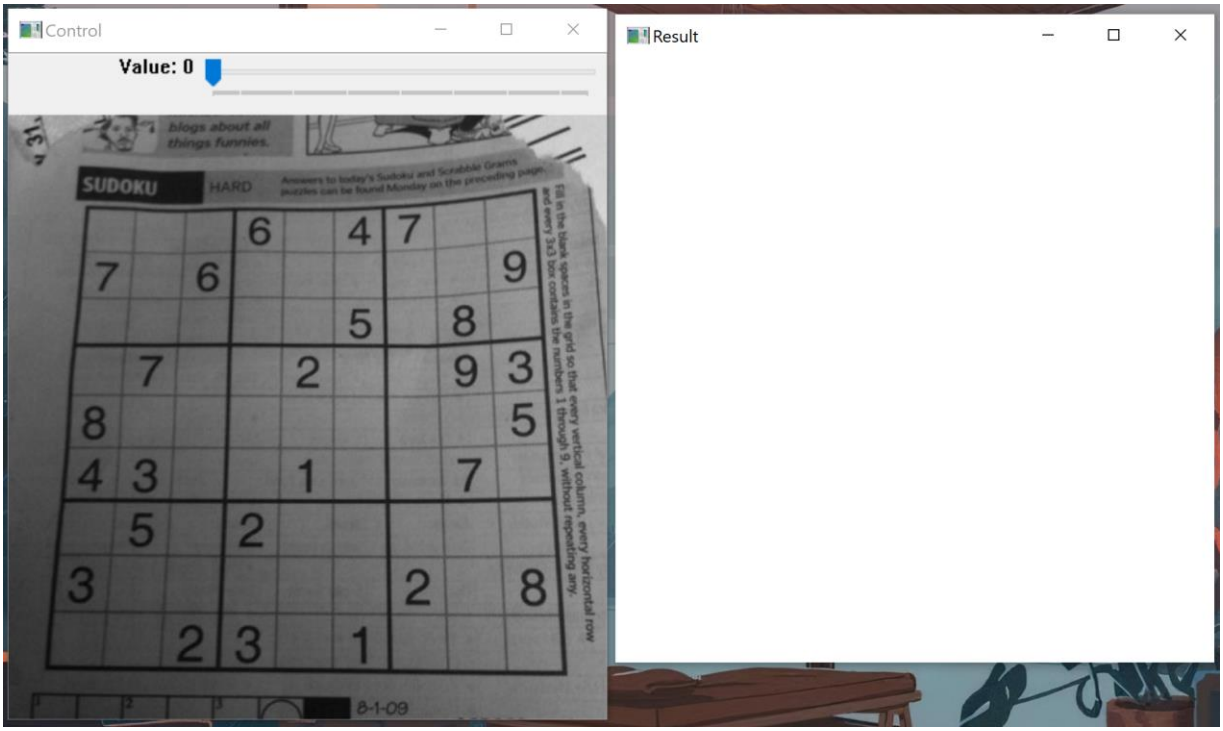
```
th, dst = cv2.threshold(source, thresholdValue, maxVal,
                        thresholdingTechnique)
```

Ở đây mình đã dùng cv2.THRESH_BINARY

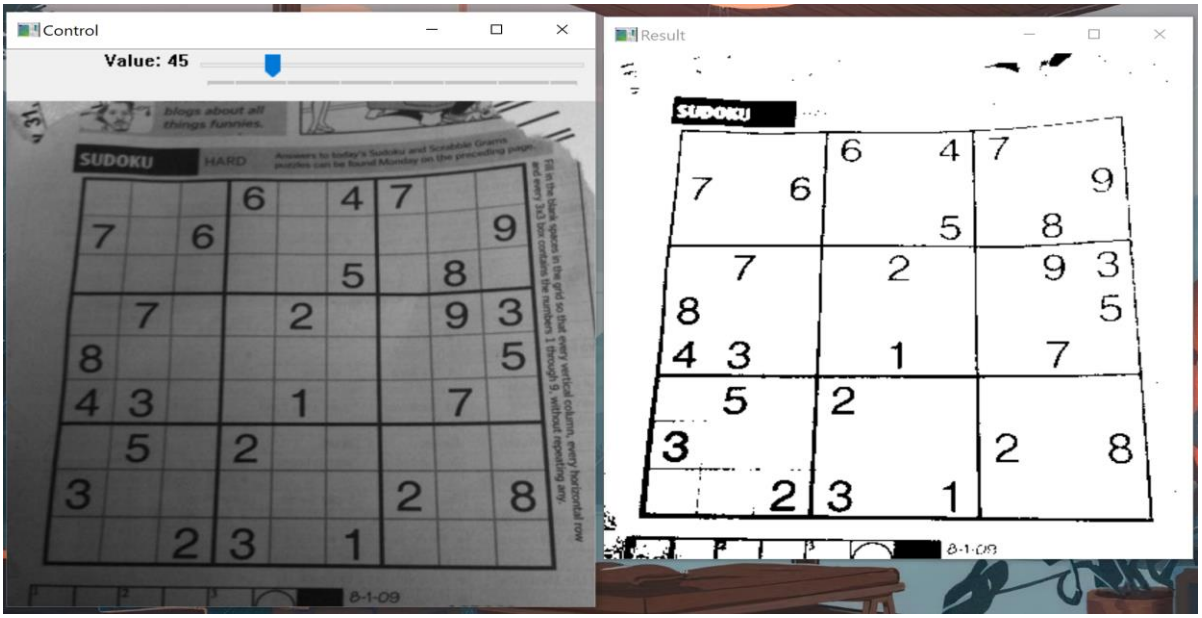
```python
import cv2

def get_value(pos):
    get_value.value=pos
get_value.value = 0

cv2.namedWindow("Control")
cv2.createTrackbar("Value", "Control", 0, 255, get_value)

while True:
    img = cv2.imread('sudoku-original.jpg',0)
    cv2.imshow("Control",img)

    ret,re = cv2.threshold(img, get_value.value,255, cv2.THRESH_BINARY )
    # print(ret)
    # re = cv2.adaptiveThreshold(img, get_value.value, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY,3,0)
    cv2.imshow("Result", re)

    if cv2.waitKey(10) == ord('q'):
        break
cv2.destroyAllWindows()
```

Ban đầu chúng ta cần khai báo thư viện. Ở đây hàm get_value là để lấy position của từng giá trị trong createTrackbar từ 0 tới 255 và tạp cái nameWindow có tên là control. Tiếp theo ta chạy vòng lặp while để lun đúng khi ta thay đổi mỗi giá trị position thì ta luôn thay đổi giá trị re (đây là kết quả của hàm chạy threshold với ngững là ta lấy từ hàm get_value). Sau đó ta sẽ cho ấn phím q để kết thúc vòng lặp và xóa tất cả màn hình.

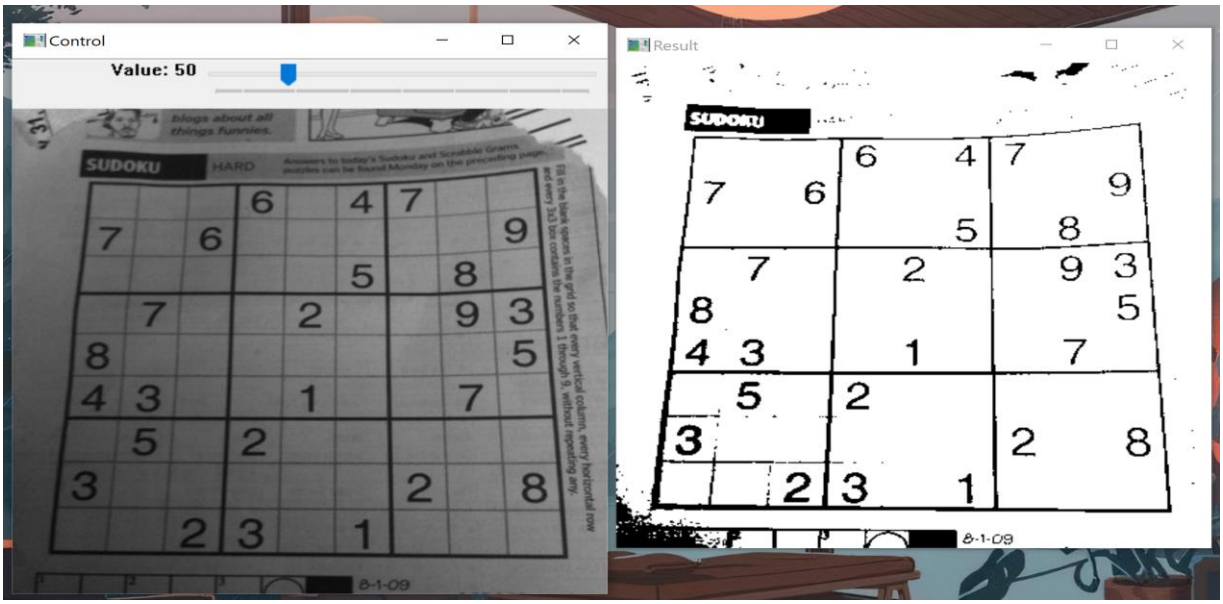
1.1.3 Nhận xét kết quả và lý do tại sao có ADAPTIVE THRESHOLDING?



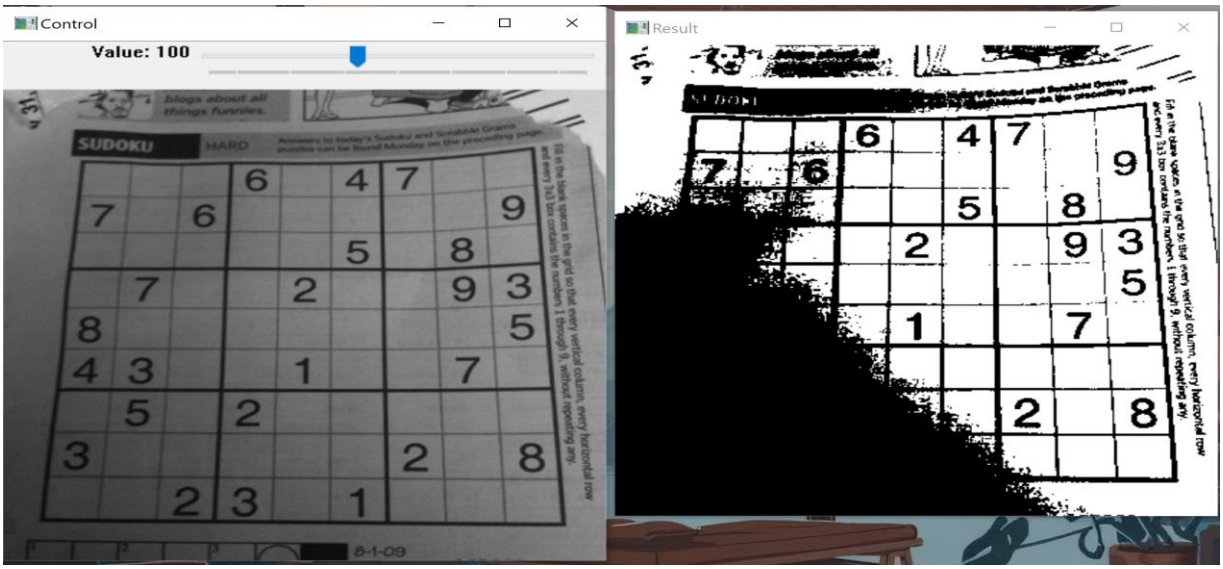
Nhìn vào input của tấm ảnh ở màn hình control ta sẽ thấy bức ảnh có mức sáng không đồng đều (phần trên bên phải sáng hơn phần dươi bên trái). Hãy để ý thanh value có giá trị 0 là ngưỡng của màn hình result.



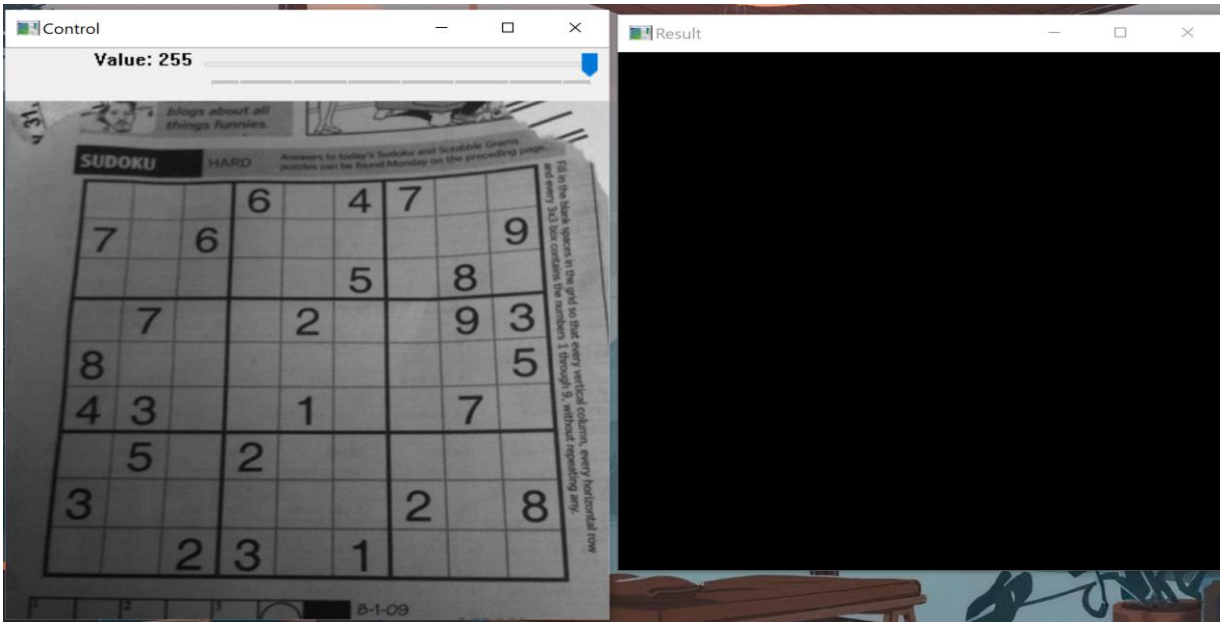
Khi ta tăng ngưỡng lên 45 thì ta sẽ có kết quả là result nhưng nó chưa thực sự tốt với cái input của sudoku. Ta sẽ tiếp tục tăng giá trị ngưỡng.



Ở giá trị 50 ta đã thấy được những màu đen bị lấn do phần ánh sáng tối bên góc trái của bức ảnh đã ảnh hưởng tới kết quả



Khi ta tăng lên giá trị 100 thì ngưỡng đã bắt đầu cao và ta hoàn toàn bị che mất phàn bên dưới góc trái đã mất thông tin. Tiếp tục tiến về 255 ta sẽ hoàn toàn có màu đen của bức ảnh.

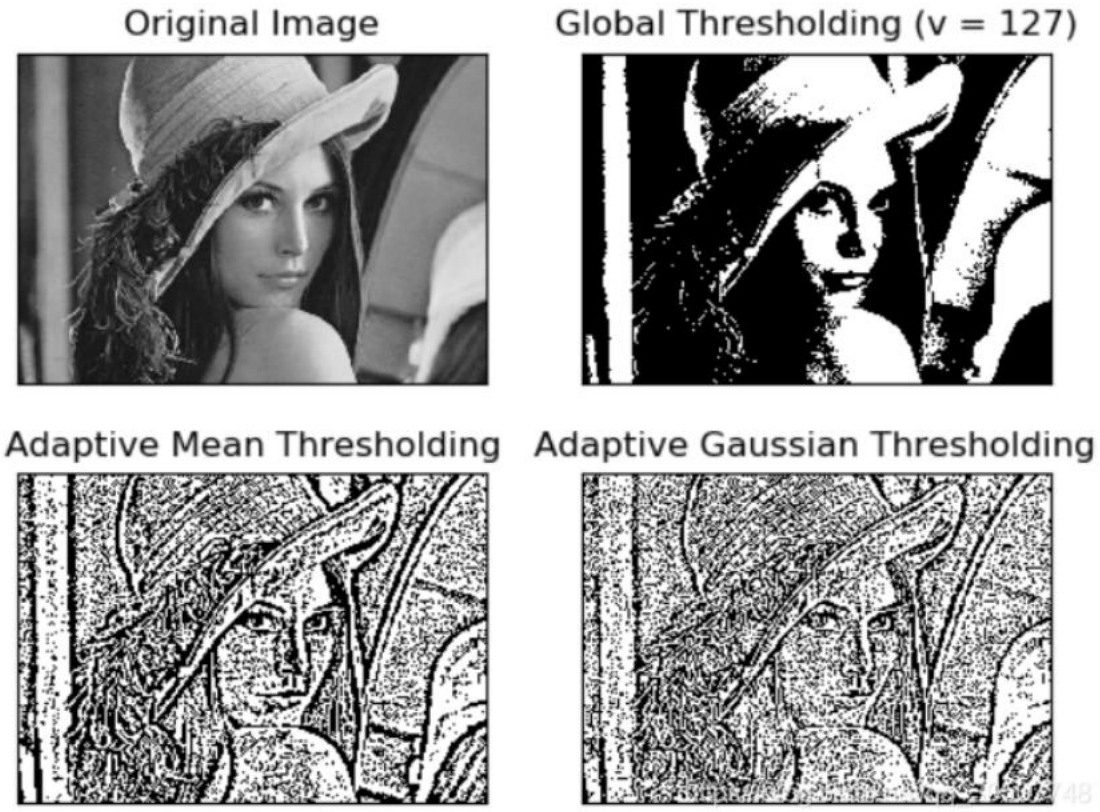


Nhìn vào ảnh thì chúng ta thấy được rằng giá trị ngường có từ giá trị từ 0 đến 255. Vấn đề ở đây là ta cần chọn ngưỡng như thế nào thị ta sẽ được tấm ảnh nhị phân như ý muốn (làm ảnh nổi các vùng đối tượng và nền). Việc xác định giá trị ngưỡng rất là khó vì chúng phụ thuộc vào từng điều kiện chiếu sáng khác nhau của môi trường. Với môi trường này nhận một giá trị ngưỡng thì môi trường khác nhận một giá trí nhưng cũng chưa chắc sẽ ra được kết quả mà mình mong muốn. Nhằm khác phục vấn đề này thì ADAPTIVE THRESHOLDING đã được tạo ra.

1.2 Adaptive thresholding là gì?

Phương pháp thresholding ở trên hoạt động khá tốt, nhưng nó không phù hợp cho nhiều trường hợp, như là ánh sáng không đồng đều, phơi sáng, bị đèn flask. Trong trường hợp đó chúng ta dùng hàm adaptiveThreshold().

Phương thức này tính giá trị trung bình của các n điểm xung quanh pixel đó rồi trừ cho C chứ không lấy ngưỡng cố định (n thường là số lẻ, còn C là số nguyên bất kỳ).



1.2.1 Những phương pháp xác định vùng.

Một trong những cách được sử dụng để giải quyết vấn đề trên là chia nhỏ bức ảnh thành những vùng nhỏ (region), và đặt giá trị ngưỡng trên những vùng nhỏ đó. Opencv cung cấp cho chúng ta hai cách xác định những vùng nhỏ.

Có 2 phương thức xác định vùng:

1. **ADAPTIVE_THRESH_MEAN_C**: giá trị của pixel phụ thuộc vào các pixel lân cận
2. **ADAPTIVE_THRESH_GAUSSIAN_C**: giá trị của pixel cũng phụ thuộc vào các pixel lân cận, tuy nhiên được khử nhiễu

CHƯƠNG 2

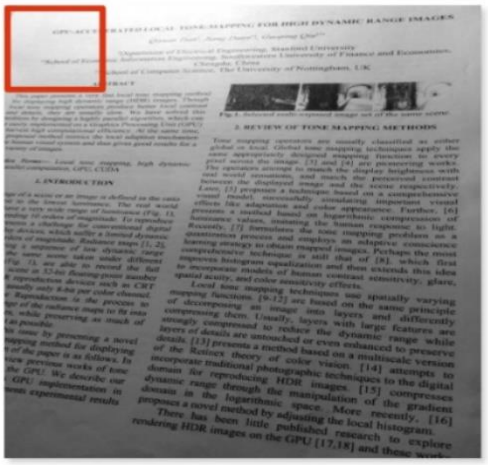
ADAPTIVE THRESHOLDING

2.1 Ý tưởng thực hiện Adaptive Thresholding

Đối với mỗi hình ảnh ta sẽ trượt từng cửa số phù hợp qua hình ảnh gọi là phân đoạn ảnh.

Đối với mỗi vị trí phân đoạn ảnh, cần phải quyết định xem có thực hiện xét ngưỡng hay không?

- Việc xác định ngưỡng không nên thực hiện ở trong các khu vực đồng nhất. Khu vực đồng nhất ở đây là khu vực ảnh được chiếu sáng đồng đều. Tức là ở đây ta sẽ có 2 trường hợp. Trường hợp thứ nhất thì đây chính là nền của bức ảnh và trường hợp thứ 2 thì phân đoạn đó thuộc đối tượng và ở 2 trường hợp này thì ánh sáng đều đồng đều nên ta sẽ không cần ngưỡng.
- Sử dụng phương sai hoặc tiêu chí phù hợp khác. Ở đây sẽ có hai phương pháp. Đầu tiên là giá trị ngưỡng là tính giá trị trung bình của khu vực lân cận. Giá trị ngưỡng là tổng trọng số khu vực lân cận (Sử dụng trong số Gauss được tính bằng phương thức getGaussiankernel())



Do việc cắt ngưỡng cho mỗi pixel phụ thuộc và vị trí của mỗi pixel lân cận trong ảnh con nên kỹ thuật này được gọi là cắt ngưỡng thích nghi (Adaptive). Hay còn gọi là cắt ngưỡng cục bộ.

2.2 Tính giá trị trung bình của khu vực lân cận:

2.2.1 Khu vực lân cận là gì?

Giả sử có điểm ảnh p tại tọa độ (x,y). p có 4 điểm gần nhất theo chiều và chiều ngang là {(x-1,y); (x,y-1); (x,y+1); (x+1,y)}.

(x-1,y-1)	(x,y-1)	(x+1,y-1)
(x-1,y)	(x,y)	(x+1,y)
(x-1, y+1)	(x,y+1)	(x+1,y+1)

Mỗi điểm ảnh sẽ có tất cả tập hợp 8 điểm lân cận của tập hợp p. Nếu điểm p nằm ở biên(mép) của ảnh, thì một số điểm sẽ nằm ngoài ảnh.

2.2.2 Cách tính giá trị trung bình của khu vực lân cận:

Nó chỉ đơn giản là cách giá trị trung bình của tất cả các pixel trong vùng phân đoạn ảnh và thay thế chúng. Đây là bộ lọc chuẩn hóa 3x3

$$K = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

2.3 Tính ngưỡng bằng tổng trọng số của khu vực lân cận

Được thực hiện bằng cách nhân chập ảnh đầu vào với một ma trận lọc Gauss sau đó cộng chúng lại để tạo thành ảnh đầu ra.

Ý tưởng chung là giá trị mỗi điểm ảnh sẽ phụ thuộc nhiều vào các điểm ảnh ở gần hơn là các điểm ảnh ở xa. Trọng số của sự phụ thuộc được lấy theo hàm Gauss.

Giả sử ảnh là một chiều.

Điểm ảnh ở trung tâm sẽ có trọng số lớn nhất. Các điểm ảnh ở càng xa trung tâm sẽ có trọng số giảm dần khi khoảng cách từ chúng tới điểm trung tâm tăng lên. Như vậy điểm càng gần trung tâm sẽ càng đóng góp nhiều hơn vào giá trị điểm trung tâm.Dưới đây là phương trình hàm Gaussian dùng trong không gian một chiều.

$$G(x) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

Hình 1: hàm Guassian trong không gian một chiều

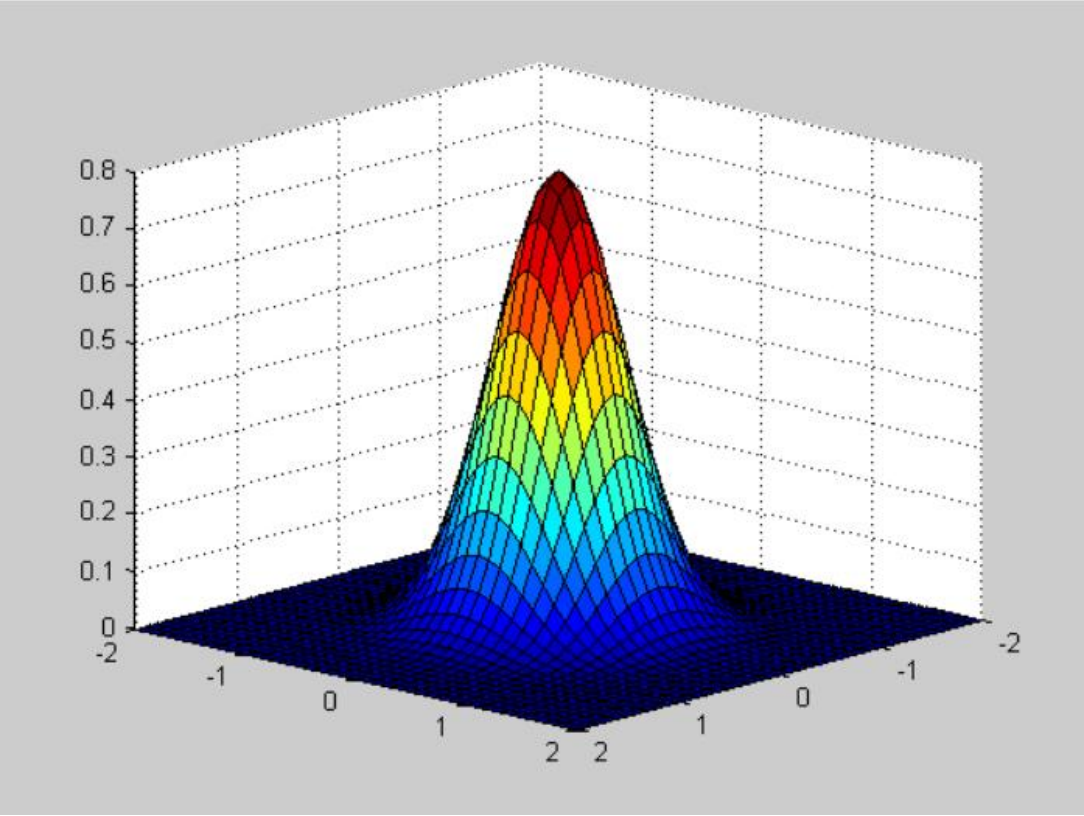
Giả sử ảnh là hai chiều.

Một hàm Gaussian 2-D nhận được bằng cách nhân hai hàm Gaussian 1-D (một cho mỗi hướng).

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Hình 2: hàm Gaussian trong không gian hai chiều.

Trong đó x và y là tọa độ theo hai trục đứng và ngang còn σ là phương sai chuẩn của phân bố Gaussian hay là giá trị quyết định độ lệch giữa các điểm trên bề mặt Gaussian.Trong phân bố Gaussian chuẩn thì giá trị kỳ vọng μ =0.



Đường cong Gauss chuẩn hóa với giá trị kỳ vọng μ = 0 và phương sai σ2 =0.2 cho mảng 2 chiều (ảnh) ; Khoảng tập trung của phân bố trong khoảng(-1.341,1.341) là 99.7%.

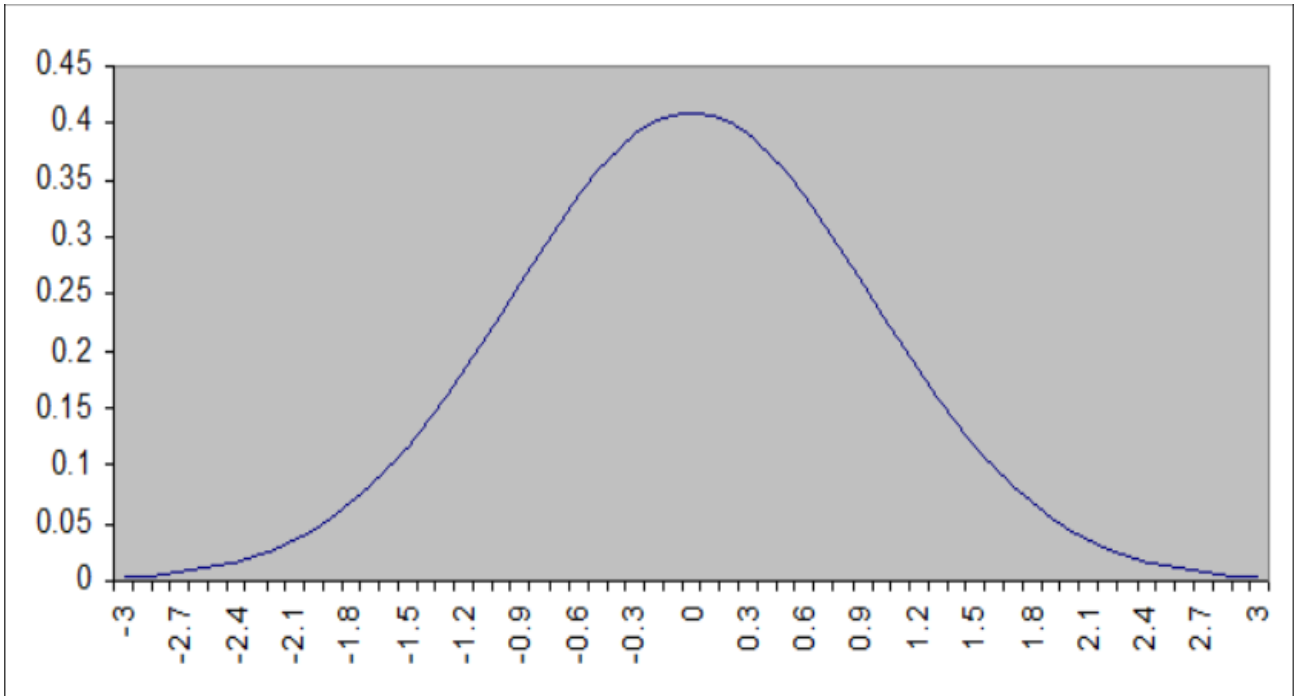
Giá trị mới của mỗi Pixel sau khi tính tích chập với nhân đại diện cho hàm Gaussian có thể coi là trung bình lượng giá trị của các pixel xung quanh nó. Tính chất này giúp giữ lại đường viền và biên cũng như làm mờ một cách đồng bộ hơn so với các phương pháp khác. Khoảng cách giữa hai điểm gần nhau trong Gaussian Kernel là σ

VD:

Bộ nhân 3 × 3 phương sai σ được openCv tính là 0.95.

$$\frac{1}{169} \times \begin{bmatrix} 6 & 26 & 6 \\ 26 & 41 & 26 \\ 6 & 26 & 6 \end{bmatrix} \quad \begin{bmatrix} 0.0355 & 0.1538 & 0.2307 \\ 0.1538 & 0.2426 & 0.1538 \\ 0.0355 & 0.1538 & 0.0355 \end{bmatrix}$$

Ma trận Gaussian 3x3



Hình 3: Biểu đồ phân bố Gaussian với phương sai σ= 0.95 của nhân 3x3

Tính chất

- Hạt nhân Gaussian có thể phân tách tuyến tính. Điều này có nghĩa là chúng ta có thể chia bất kỳ bộ lọc 2-D nào thành hai bộ lọc 1-D. Do đó, độ phức tạp tính toán được giảm từ O (n^2) xuống O (n). Ví dụ

$$\frac{1}{169}\begin{pmatrix} 6 & 26 & 6 \\ 26 & 41 & 26 \\ 6 & 26 & 6 \end{pmatrix} = \frac{1}{169}\begin{pmatrix} 6 \\ 26 \\ 6 \end{pmatrix}\begin{pmatrix} 6 & 26 & 6 \end{pmatrix}$$

- Việc áp dụng nhiều hạt nhân Gaussian liên tiếp tương đương với việc áp dụng một vệt mờ Gaussian đơn lẻ, lớn hơn, có bán kính là căn bậc hai của tổng bình phương của bán kính nhiều hạt nhân. Sử dụng thuộc tính này, chúng tôi có thể ước lượng một bộ lọc không thể phân tách bằng sự kết hợp của nhiều bộ lọc có thể phân tách.

Do những đặc tính này, Gaussian Blurring là một trong những thuật toán hiệu quả nhất và được sử dụng rộng rãi.

2.4 Tìm ngưỡng thích ứng với một ma trận điểm ảnh

Giả sử, có ảnh I(m*n)

Với G là các điểm mức xám của ảnh, và sắp xếp tăng dần

t(g): số lần xuất hiện của các mức xám <=g

h(g): số lần xuất hiện của các mức xám

+ Momen quán tính trung bình có mức xám <= g:

$$m(g) = \frac{1}{t(g)} \sum_0^g g.h(g)$$

+ Hàm f:

$$f(g) = \frac{t(g)}{m*n - t(g)} [m(g) - m(G-1)]^2$$

➔ Ngưỡng tự động: Θ

➔ f(Θ) = max(f(g))

10	11	15	7
9	10	11	15
9	9	10	11
5	9	9	10

g	5	7	9	10	11	15
h(g)	1	1	5	4	3	2
t(g)	1	2	7	11	14	16
g*h(g)	5	7	45	40	33	30
m(g)	5	6	57/7	97/11	65/7	10
f(g)	1.67	2.29	2.68	3.07	3.57	∞

2.5 Code minh họa.

test1.py > ...

```python
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
img = cv.imread('sudoku.png',0)
img = cv.medianBlur(img,5)
ret,th1 = cv.threshold(img,127,255,cv.THRESH_BINARY)
th2 = cv.adaptiveThreshold(img,255,cv.ADAPTIVE_THRESH_MEAN_C,cv.THRESH_BINARY,11,2)
th3 = cv.adaptiveThreshold(img,255,cv.ADAPTIVE_THRESH_GAUSSIAN_C, cv.THRESH_BINARY,11,2)
titles = ['Original Image', 'Global Thresholding (v = 127)', 'Adaptive Mean Thresholding', 'Adaptive Gaussian Thresholding']

images = [img, th1, th2, th3]
for i in np.arange(4):
    plt.subplot(2,2,i+1), plt.imshow(images[i],'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
plt.show()
```

Ở đây:

maxValue: Đây là giá trị được gán cho các pixel sau khi tạo ngưỡng.

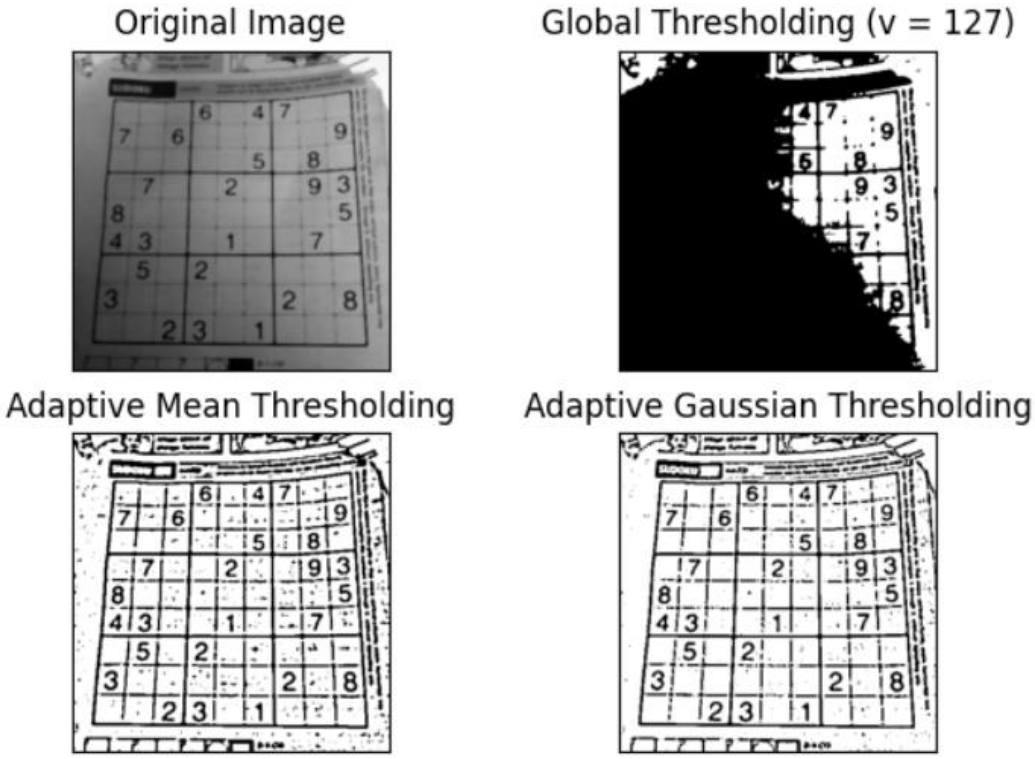
adaptiveMethod: hai giá trị là cv.ADAPTIVE_THRESH_MEAN_C và cv.ADAPTIVE_THRESH_GAUSSIAN_C, đó là các phương pháp tính ngưỡng.

thresholdType: Tương tự như Simple Thresholding đã trình bày ở trên.

BlockSize: số pixel lân cận dùng để tính toán, phải là một số lẻ lớn hơn 0.

C : hằng số trừ đi giá trị trung bình, từ -255 đến 255.

2.6 Nhận xét kết quả.



Như chúng ta đã phân tích ở phần Simple thresholding ảnh gốc có mức sáng không đồng đều, khi cho ngưỡng 127 thì những màu đen bị lấn do phần ánh sáng tối bên góc trái của bức ảnh đã ảnh hưởng tới kết quả, việc xác định được giá trị threshold phù hợp là không đơn giản và yêu cầu nhiều kinh nghiệm.

Về cơ bản thì chúng ta có thể thấy bộ lọc Gaussian giúp giảm nhiễu tốt hơn bộ lọc Mean, điều này là vì bộ lọc Gaussian cân các pixel thành một đường cong hình chuông xung quanh pixel trung tâm. Có nghĩa là các pixel xa hơn sẽ có trọng số thấp hơn.

Bộ lọc Mean chỉ tính trung bình các giá trị pixel của tất cả các pixel lân cận. Điều này tương đương với việc cung cấp một trọng lượng bằng nhau cho tất cả các pixel xung quanh trung tâm bất kể khoảng cách từ pixel trung tâm.

2.7 Kết luận:

Cũng không có gì đảm bảo rằng ngưỡng do hàm tự động tính toán là tốt nhất. Trên thực tế, không có một tiêu chuẩn nào để đánh giá ngưỡng đó là tốt hay không.

Bản chất của ngưỡng thích ứng vẫn là phân đoạn ngưỡng, nghĩa là giá trị lớn nhất được lấy khi giá trị ngưỡng bị vượt quá và giá trị nhỏ nhất được lấy nếu không đạt đến giá trị ngưỡng. Chỉ là nó có thể tự động tính toán giá trị tới hạn trong một khu vực nhỏ. Điều đó có nghĩa là, mặc dù nó có thể tính toán ngưỡng một cách thông minh, vẫn không thể thoát khỏi khiếm khuyết cơ bản của ngưỡng.

TÀI LIỆU THAM KHẢO

1. https://www.youtube.com/watch?v=B1ocEhpx0TM
2. https://www.researchgate.net/publication/335080661_A_threshold_segmentation_method_for_nonuniform_illumination_image_based_on_brightness_equalization?fbclid=IwAR221tshBSZd3tKOZL9LCRdzDSZNjTxZaglOYaFYaeum3vRkSwJ0ypDT74
3. https://theailearner.com/2019/07/20/adaptivethresholding/?fbclid=IwAR3FSAKEM3jCNYf992_iPQnarIMt7QMDOBesbUlOwnoYX15woVg-XrworEs
4. https://theailearner.com/2019/07/19/improving-globalthresholding/?fbclid=IwAR2G0G2CVYTceU9ZEw0OYDm6v2FR3nAJe1wQJlgcBJ3kiNNrMGFMHCcsc
5. https://nguyenkhoaninh.wordpress.com/2014/04/20/opencv-nhi-phan-hoa-anh-voi-nguong-dong-adaptive-threshold/?fbclid=IwAR1JkKkjb5vZW-m7PUiyCVxyJDS7Rxc4hgTy3UUsP7exM7841Ii1Z_czreI
6. https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html?fbclid=IwAR3xWSqAYsbPYO-x9kPXgyYV4XjrM5iQunm1KSmP-tDvGwVOapwfC_xXdRM
7. https://chercher.tech/opencv/image-threshold-opencv