**TRƯỜNG ĐẠI HỌC BÁCH KHOA THÀNH PHỐ HỒ CHÍ MINH**



**NHẬP MÔN THỊ GIÁC MÁY TÍNH**

**Bài tập Introduction to camera và điểm ảnh**

GVHD : TS. Lê Đức Hạnh

NHÓM : 7

|  |  |  |
| --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** |
| 1 | Phan Trương Minh Khánh | 1812602 |
| 2 | Huỳnh Minh Khánh | 1812585 |
| 3 | Võ Văn Thiên Phú | 1811146 |
| 4 | Hoàng Minh Phong | 1811141 |
| 5 | Mã Tiến Phúc | 1810441 |

**TP.HCM, tháng 9 năm 2021**

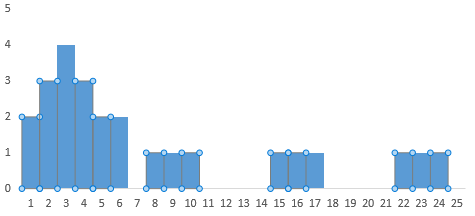
**Bài 1:**

Cho hình có ma trận sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 20 | 30 | 40 | 10 | 20 |
| 10 | 80 | 90 | 100 | 60 |
| 50 | 150 | 160 | 170 | 30 |
| 30 | 220 | 230 | 240 | 50 |
| 40 | 60 | 30 | 40 | 20 |

**Bài làm:**

a. Vẽ histogram của hình bên bằng tay và tiến hành phân ngưỡng để thấy vùng sáng bằng tay.



b.

Đếm số lần xuất hiện của các số

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Pixel intensity | No. of pixel | Probability | C.P | C.P\*255 | Chọn cho 255 | C.P\*50 |
| 10 | 2 | 0.08 | 0.08 | 20.4 | 20 | 4 |
| 20 | 3 | 0.12 | 0.20 | 51 | 51 | 10 |
| 30 | 4 | 0.16 | 0.36 | 91.8 | 92 | 18 |
| 40 | 3 | 0.12 | 0.48 | 122.4 | 122 | 24 |
| 50 | 2 | 0.08 | 0.56 | 142.8 | 143 | 28 |
| 60 | 2 | 0.08 | 0.64 | 163.2 | 163 | 32 |
| 80 | 1 | 0.04 | 0.68 | 173.4 | 173 | 34 |
| 90 | 1 | 0.04 | 0.72 | 183.6 | 184 | 36 |
| 100 | 1 | 0.04 | 0.76 | 193.8 | 194 | 38 |
| 150 | 1 | 0.04 | 0.80 | 204 | 204 | 40 |
| 160 | 1 | 0.04 | 0.84 | 214.2 | 214 | 42 |
| 170 | 1 | 0.04 | 0.88 | 224.4 | 224 | 44 |
| 220 | 1 | 0.04 | 0.92 | 234.6 | 235 | 46 |
| 230 | 1 | 0.04 | 0.96 | 244.8 | 245 | 48 |
| 240 | 1 | 0.04 | 1.00 | 255 | 255 | 50 |

- Ma trận trên là ma trận 5x5, cho mỗi điểm ảnh với pixel là 1.

- Tính xác suất của một điểm ảnh xuất hiện:

Điểm 10 xuất hiện tổng cộng là 2 lần trong hình.

Xác suất xuất hiện của điểm 10:

Tương tự với các điểm ảnh khác:

Tính C.P:

Cân bằng sáng sử dụng 256 bin, tính lại pixel thay thế:

, chọn

, chọn

, chọn

Ta được ma trận sau:

256 bins

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 51 | 92 | 122 | 20 | 51 |
| 20 | 173 | 184 | 194 | 163 |
| 143 | 204 | 214 | 224 | 92 |
| 92 | 235 | 245 | 255 | 143 |
| 122 | 163 | 92 | 122 | 51 |

50 bins:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | 18 | 24 | 4 | 10 |
| 4 | 34 | 36 | 38 | 32 |
| 28 | 40 | 42 | 44 | 18 |
| 18 | 46 | 48 | 50 | 28 |
| 24 | 32 | 18 | 24 | 10 |

c.

Code:

|  |
| --- |
| #include <opencv2/opencv.hpp>  #include <iostream>  #include<stdio.h>  #include<stdlib.h>  #include <stdint.h>  using namespace cv;  using namespace std;  void histogram(string const& name, Mat const& Image)  {  int bin = 256;  int histsize[] = { bin };  float range[] = { 0, 255 };  const float\* ranges[] = { range };  Mat hist;  int chanel[] = { 0 };  int hist\_heigt = 256;  Mat hist\_image = Mat::zeros(hist\_heigt, bin, CV\_8SC3);  calcHist(&Image, 1, chanel, Mat(), hist, 1, histsize, ranges, true, false);  double max\_val = 0;  minMaxLoc(hist, 0, &max\_val);  for (int i = 0; i < bin; i++)  {  float binV = hist.at<float>(i);  int a = binV \* hist\_heigt;  int height = cvRound(a / max\_val);  line(hist\_image, Point(i, hist\_heigt - height), Point(i, hist\_heigt), Scalar::all(255));  }  imshow(name, hist\_image);  }  int main(int argv, char\*\* argc)  {  float anh[25] = { 20, 30, 40, 10, 20,  10, 80, 90, 100, 60,  50, 150, 160, 170, 30,  30, 220, 230, 240, 50,  40, 60, 30, 40, 20 };  Mat gray\_anh = Mat(5, 5, CV\_32F, anh);  Mat gray\_anh\_his;  gray\_anh.convertTo(gray\_anh, CV\_8UC1);  equalizeHist(gray\_anh, gray\_anh\_his);  cout << "matrix = " << endl << "" << gray\_anh << endl << endl;  cout << "Matrix\_hist\_256 = " << endl << "" << gray\_anh\_his << endl << endl;  cout << "Matrix\_hist\_50 = " << endl << "" << gray\_anh\_his \*49/255 << endl << endl;  histogram("Hist\_old", gray\_anh);  histogram("Hist\_new", gray\_anh\_his);  waitKey(0);  } |
|  |

**Bài 3:** Cho 2 hình sau:



a) load 2 hình và tính khoảng cách Euclid của 2 màu (R,G,B), (nhớ đưa về cùng kích thước)

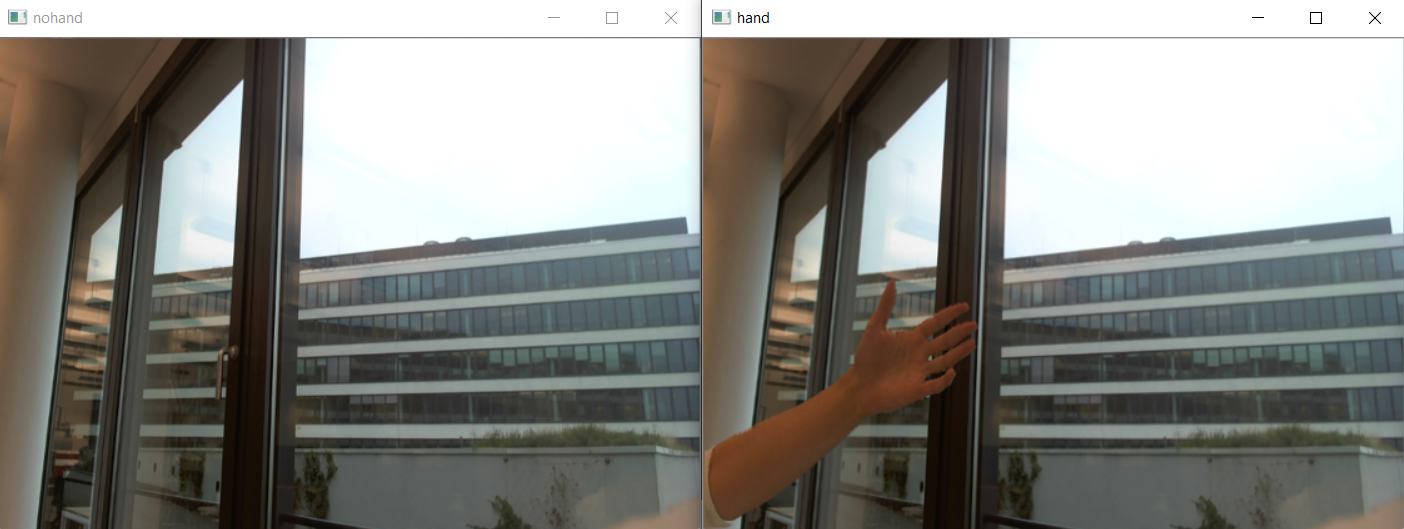
b) Set threshold để thấy cánh tay.

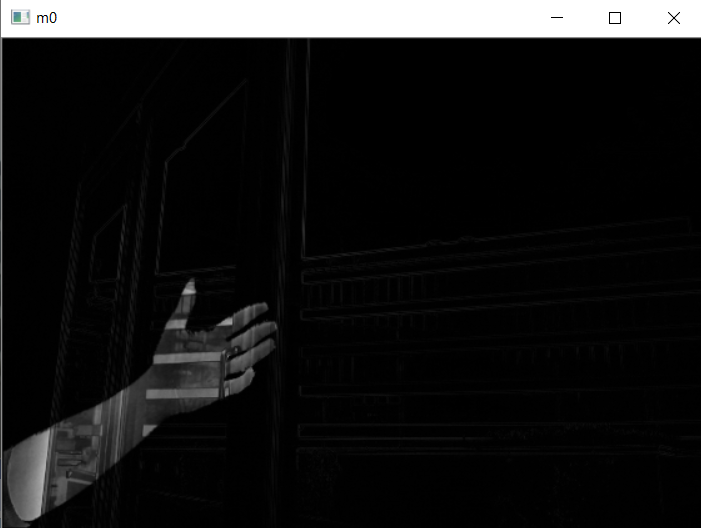
**Bài làm:**

1. load 2 hình và tính khoảng cách Euclid của 2 màu (R,G,B).( cho 2 hình về cùng kích thước)

|  |
| --- |
| //Khai báo thư viện:  #include <opencv2/core/core.hpp>  #include <opencv2/video/video.hpp>  #include <opencv2/imgproc.hpp>  #include <opencv2/highgui/highgui.hpp>  #include <opencv2/imgcodecs.hpp>  #include <iostream>  using namespace cv;  using namespace std;  int main(int argc, char\*\* argv)  {  // Load 2 hình, một hình không có cánh tay và một hình có:  Mat nohand = imread("nohand.png");  Mat hand = imread("hand.png");  // Tạo ma trận zero  Mat m0 = Mat::zeros(nohand.rows, nohand.cols, CV\_8UC1);  // Resize 2 hình:  resize(hand, hand, Size(nohand.size().width, nohand.size().height));  int hang = nohand.rows;  int cot = nohand.cols;  float euclid\_dis;  // Tính khoảng cánh Euclid  for (int i = 0; i < hang; i++) {  for (int j = 0; j < cot; j++) {  Vec3b a = nohand.at<Vec3b>(i, j);  Vec3b b = hand.at<Vec3b>(i, j);  euclid\_dis = sqrt(((a[0] - b[0]) \* (a[0] - b[0])) + ((a[1] - b[1]) \* (a[1] - b[1])) + ((a[2] - b[2]) \* (a[2] - b[2])));  m0.at<uint8\_t>(i, j) = euclid\_dis;  }  }  // Show 3 ảnh:  imshow("nohand", nohand);  imshow("hand", hand);  imshow("m0", m0);  waitKey(0);  return 0;  } |

**Kết quả:**

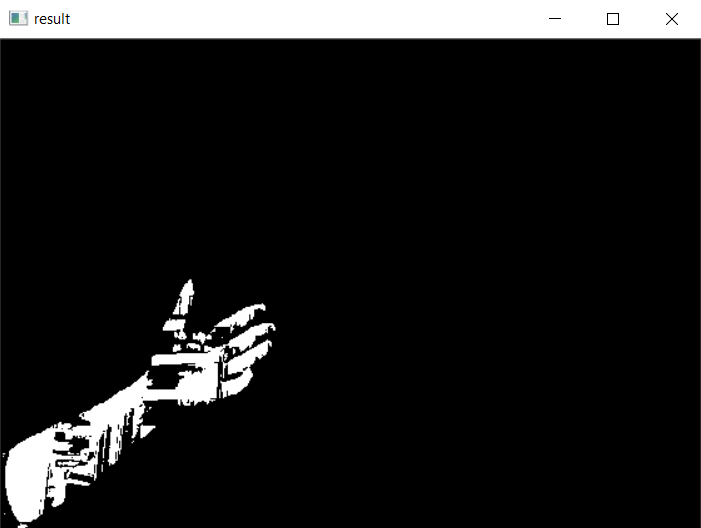




b) Set threshold để thấy cánh tay:

|  |
| --- |
| //Khai báo thư viện:  #include <opencv2/core/core.hpp>  #include <opencv2/video/video.hpp>  #include <opencv2/imgproc.hpp>  #include <opencv2/highgui/highgui.hpp>  #include <opencv2/imgcodecs.hpp>  #include <iostream>  using namespace cv;  using namespace std;  int main(int argc, char\*\* argv)  {  // Load 2 hình, một hình không có cánh tay và một hình có:  Mat nohand = imread("nohand.png");  Mat hand = imread("hand.png");  // Tạo ma trận zero:  Mat m0 = Mat::zeros(nohand.rows, nohand.cols, CV\_8UC1);  // Resize 2 hình:  resize(hand, hand, Size(nohand.size().width, nohand.size().height));  int hang = nohand.rows;  int cot = nohand.cols;  float euclid\_dis;  //Set giá trị threshold để thấy cánh tay:  float threshold = 47;  //Tính khoảng cách euclid  for (int i = 0; i < hang; i++) {  for (int j = 0; j < cot; j++) {  Vec3b a = nohand.at<Vec3b>(i, j);  Vec3b b = hand.at<Vec3b>(i, j);  euclid\_dis = sqrt(((a[0] - b[0]) \* (a[0] - b[0])) + ((a[1] - b[1]) \* (a[1] - b[1])) + ((a[2] - b[2]) \* (a[2] - b[2])));  //Lưu điểm ảnh được tăng sáng vào ma trận zero:  if (euclid\_dis > threshold)  {  m0.at<unsigned char>(i, j) = 255;  }  }  }  //Hiển thị hình ảnh cánh tay:  imshow("result", m0);  waitKey(0);  return 0;  } |

**Kết quả:**



**Bài 4:**

Cho Ma trận sau:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | STT | 40 | 5 | 6 | 7 |
| 8 | STT | STT+2 | 11 | 40 | 13 | 14 |
| 11 | 40 | 40 | 40 | STT-1 | 40 | STT+2 |
| 8 | STT | 40 | 10 | STT | 10 | 15 |
| 10 | 2 | 4 | 40 | 10 | 7 | 8 |
| 1 | 2 | STT | 4 | 5 | STT | 7 |

1. Tính bằng tay local threshold với window là 3x3 bằng phương pháp tính giá trị trung bình của window mà nó trượt (thống nhất thêm padding 0 và một lần trượt 1 cột)
2. Lập trình tính local threshold sử dùng hàm adaptive threshold và in ra kết quả

**Bài làm:**

STT = 7

1. Thêm padding 0 cho ma trận

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 2 | 7 | 40 | 5 | 6 | 7 | 0 |
| 0 | 8 | 7 | 9 | 11 | 40 | 13 | 14 | 0 |
| 0 | 11 | 40 | 40 | 40 | 6 | 40 | 9 | 0 |
| 0 | 8 | 7 | 40 | 10 | 7 | 10 | 15 | 0 |
| 0 | 10 | 2 | 4 | 40 | 10 | 7 | 8 | 0 |
| 0 | 1 | 2 | 7 | 4 | 5 | 7 | 7 | 0 |

* Tính giá trị vị trí hàng 1 cột 1:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 2 | 7 | 40 | 5 | 6 | 7 | 0 |
| 0 | 8 | 7 | 9 | 11 | 40 | 13 | 14 | 0 |
| 0 | 11 | 40 | 40 | 40 | 6 | 40 | 9 | 0 |
| 0 | 8 | 7 | 40 | 10 | 7 | 10 | 15 | 0 |
| 0 | 10 | 2 | 4 | 40 | 10 | 7 | 8 | 0 |
| 0 | 1 | 2 | 7 | 4 | 5 | 7 | 7 | 0 |

Giá trị threshold:

* Giá trị ở ô này sau khi threshold là 0
* Tính giá trị vị trí hàng 1 cột 2:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 2 | 7 | 40 | 5 | 6 | 7 | 0 |
| 0 | 8 | 7 | 9 | 11 | 40 | 13 | 14 | 0 |
| 0 | 11 | 40 | 40 | 40 | 6 | 40 | 9 | 0 |
| 0 | 8 | 7 | 40 | 10 | 7 | 10 | 15 | 0 |
| 0 | 10 | 2 | 4 | 40 | 10 | 7 | 8 | 0 |
| 0 | 1 | 2 | 7 | 4 | 5 | 7 | 7 | 0 |

Giá trị threshold:

* Giá trị ở ô này sau khi threshold là 0
* Tính giá trị vị trí hàng 1 cột 3:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 2 | 7 | 40 | 5 | 6 | 7 | 0 |
| 0 | 8 | 7 | 9 | 11 | 40 | 13 | 14 | 0 |
| 0 | 11 | 40 | 40 | 40 | 6 | 40 | 9 | 0 |
| 0 | 8 | 7 | 40 | 10 | 7 | 10 | 15 | 0 |
| 0 | 10 | 2 | 4 | 40 | 10 | 7 | 8 | 0 |
| 0 | 1 | 2 | 7 | 4 | 5 | 7 | 7 | 0 |

Giá trị threshold:

* Giá trị ở ô này sau khi threshold là 0
* Tương tự với các ô khác ta có bảng sau:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

1. Đoạn code dùng để xử lý adaptive threshold:

#include "opencv2/core/core.hpp"

#include "opencv2/highgui/highgui.hpp"

#include "opencv2/imgproc/imgproc.hpp"

#include "iostream"

using namespace cv;

using namespace std;

//This code is for redefine some enums

enum {

CV\_ADAPTIVE\_THRESH\_MEAN\_C = 0,

CV\_ADAPTIVE\_THRESH\_GAUSSIAN\_C = 1,

CV\_BGR2GRAY = 6

};

void main() {

//Input data as a 2D integer matrix

int data[6][7] = { {1,2,7,40,5,6,7}, {8,7,9,11,40,13,14},{11,40,40,40,6,40,9}, { 8,7,40,10,7,10,15}, {10,2,4,40,10,7,8}, {1,2,7,4,5,7,7} };

//Initialize 2D integer matrix

cv::Mat dst(6,7, CV\_8UC1), src(6,7, CV\_8UC1);

int i, j;

//Input data from 2D int matrix to Mat

for (i = 0; i < src.rows; i++) {

for (j = 0; j < src.cols; j++) {

src.at<uchar>(i, j) = data[i][j];

}

}

//Print Mat data to screen

std::cout << src << endl<<endl;

//Threshold Mat's data

cv::adaptiveThreshold(src, dst, 1, ADAPTIVE\_THRESH\_MEAN\_C, THRESH\_BINARY, 3, 0);

//Print Threshold matrix to screen

std::cout << dst << endl;

waitKey(0);

}

Kết quả từ code so với tính tay là giống nhau:

Text

Description automatically generated

***Hình 4.1:*** Kết quả sau khi cân bằng ảnh

**Bài 5:** Cho Ma trận

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 20 | 30 | 40 | 10 | 20 |
| 10 | 80 | 90 | 100 | 60 |
| 50 | 150 | 160 | 170 | 30 |
| 30 | 220 | 230 | 240 | 50 |
| 40 | 60 | 30 | 40 | 20 |

a) Phân ngưỡng ma trận hình bằng tay sử dụng phương pháp Otsu (1.5đ)

b) Lập trình tính threshold sử dùng hàm threshold opencv với phương pháp Otsu và in ra kết quả (1đ)

Bài làm:

Ta có ma trận:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 20 | 30 | 40 | 10 | 20 |
| 10 | 80 | 90 | 100 | 60 |
| 50 | 150 | 160 | 170 | 30 |
| 30 | 220 | 230 | 240 | 50 |
| 40 | 60 | 30 | 40 | 20 |

1. Phân ngưỡng ma trận hình bằng tay sử dụng phương pháp Otsu

Khởi tạo giá trị cho T:

Phân hoạch ma trận sử dụng T1 (với nhóm G1 màu đen và G2 màu xanh):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 20 | 30 | 40 | 10 | 20 |
| 10 | 80 | 90 | 100 | 60 |
| 50 | 150 | 160 | 170 | 30 |
| 30 | 220 | 230 | 240 | 50 |
| 40 | 60 | 30 | 40 | 20 |

Tính giá trị trung bình m1 và m2 của các điểm ảnh thuộc G1 và G2:

Tính lại giá trị T:

Phân hoạch ma trận sử dụng T2:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 20 | 30 | 40 | 10 | 20 |
| 10 | 80 | 90 | 100 | 60 |
| 50 | 150 | 160 | 170 | 30 |
| 30 | 220 | 230 | 240 | 50 |
| 40 | 60 | 30 | 40 | 20 |

Do các nhóm G1 và G2 của ma trận không có sự thay đổi nên chọn T = 118 và kết quả threshold như sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 255 | 255 | 255 | 0 |
| 0 | 255 | 255 | 255 | 0 |
| 0 | 0 | 0 | 0 | 0 |

1. Code và kết quả in ra màn hình:

|  |
| --- |
| #include <opencv2/highgui.hpp>  #include <opencv2/core.hpp>  #include <opencv2/imgproc.hpp>  #include <iostream>  using namespace cv;  using namespace std;  int main(){  //KHOI TAO MA TRAN  int mtr[5][5] ={{20,30,40,10,20},  {10,80,90,100,60},  {50,150,160,170,30},  {30,220,230,240,50},  {40,60,30,40,20}};  Mat src(5, 5, CV\_8UC1);  int i, j;  cout << "MA TRAN BAN DAU:" << endl;  for (i = 0; i < 5; i++) {  for (j = 0; j < 5; j++) {  src.at<uint8\_t>(i, j) = mtr[i][j];  }  }  //IN MA TRAN RA MAN HINH  for (i = 0; i < 5; i++) {  for (j = 0; j < 5; j++) {  cout << "\t" << (int)src.at<uint8\_t>(i, j);  }  cout << endl;  }  //TINH T THEO PHUONG PHAP OTSU  Mat dst;  double thresh = 0;  double maxValue = 255;  long double thres = threshold(src, dst, thresh, maxValue, THRESH\_OTSU);  cout << "T = " << thres << endl;  //THUC HIEN THRESHOLD  for (i = 0; i < 5; i++) {  for (j = 0; j < 5; j++) {  if (src.at<uint8\_t>(i, j) > thres)  src.at<uint8\_t>(i, j) = 255;  else  src.at<uint8\_t>(i, j) = 0;  }  }  //KET QUA THRESHOLD  cout << "MA TRAN SAU KHI THRESHOLD THEO PP OTSU:" << endl;  for (i = 0; i < 5; i++) {  for (j = 0; j < 5; j++) {  cout << "\t" << (int)src.at<uint8\_t>(i, j);  }  cout << endl;  }  return 0;  } |
|  |