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



**NHẬP MÔN THỊ GIÁC MÁY TÍNH**

**Bài tập Morphological operation 1**

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 10 năm 2021**

**Bài 1:**

**Cho hình có ma trận sau:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **STT** | **10** | **11** |
| **STT** | **10** | **11** | **16** | **STT+8** | **18** |
| **16** | **STT+8** | **18** | **22** | **24** | **25** |
| **23** | **24** | **25** | **25** | **4** | **7** |

**Template:**

|  |  |  |
| --- | --- | --- |
| **STT** | **10** | **11** |
| **16** | **STT+8** | **18** |
| **23** | **24** | **25** |

**a) Tìm hình gần giống hình mẫu bằng tay sử dụng Normalized Cross-Correlation. Với Threshold > 0.98**

**b) Sử dụng hàm template matching của opencv để so sánh với kết quả câu a và b, Với Threshold > 0.98, hiện thị tọa độ của điểm tìm được**

**Bài làm**

a) Ta có ma trận ban đầu (I1) (STT = 7):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 7 | 10 | 11 |
| 7 | 10 | 11 | 16 | 15 | 18 |
| 16 | 15 | 18 | 22 | 24 | 25 |
| 23 | 24 | 25 | 25 | 4 | 7 |

Template (I2):

|  |  |  |
| --- | --- | --- |
| 7 | 10 | 11 |
| 16 | 15 | 18 |
| 23 | 24 | 25 |

- Gọi ma trận ban đầu (W x H), và Template (w x h), khi sử dụng Normalized Cross Correlation, kết quả cho ra sẽ là ma trận [(W – w + 1) x (H – h +1)], tức là kết quả sẽ là ma trận 2x4 và dưới dạng nhị phân.

- Ta có công thức:

Với: s – thành phần của ma trận sau normalize cross correlation.

I1[u, v] – thành phần của ma trận I1.

I2[u, v] – thành phần của ma trận I2.

- Tính:

Ta có :

Áp dụng công thức tính các phần tử ma trận kết quả như sau:

- Vậy ta có kết quả là:

|  |  |  |  |
| --- | --- | --- | --- |
| 0.9715 | 0.9815 | 0.9905 | 0.99985 |
| 1 | 0.9935 | 0.9029 | 0.8264 |

=> Tọa độ các điểm > 0.98 trong ma trận kết quả là 5 điểm bao gồm:

Nếu lấy điểm đầu ( 0.9715 ) là tọa độ ( 1, 1) thì ta có các tọa độ điểm > 0.98 như sau:

P1(1,2) ; P2(1,3) ; P3(1,4) ; P4(2,1) ; P5(2,2)

=> Vậy ta có 4 ma trận gần giống với hình mẫu với Threshold > 0.98 và 1 ma trận giống y như hình mẫu, đó là các hình được tô đậm sau đây ứng với 5 điểm trung tâm > 0.98 ta tìm được trên ma trận 2x4.

Hình 1: P1(1,2) giá trị P = 0.9815

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 7 | 10 | 11 |
| 7 | 10 | 11 | 16 | 15 | 18 |
| 16 | 15 | 18 | 22 | 24 | 25 |
| 23 | 24 | 25 | 25 | 4 | 7 |

Hình 2: P1(1,3) giá trị P = 0.9905

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 7 | 10 | 11 |
| 7 | 10 | 11 | 16 | 15 | 18 |
| 16 | 15 | 18 | 22 | 24 | 25 |
| 23 | 24 | 25 | 25 | 4 | 7 |

Hình 3: P1(1,4) giá trị P = 0.99985

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 7 | 10 | 11 |
| 7 | 10 | 11 | 16 | 15 | 18 |
| 16 | 15 | 18 | 22 | 24 | 25 |
| 23 | 24 | 25 | 25 | 4 | 7 |

Hình 4: P4(2,1) giá trị P = 1 ( hình giống nhất )

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 7 | 10 | 11 |
| 7 | 10 | 11 | 16 | 15 | 18 |
| 16 | 15 | 18 | 22 | 24 | 25 |
| 23 | 24 | 25 | 25 | 4 | 7 |

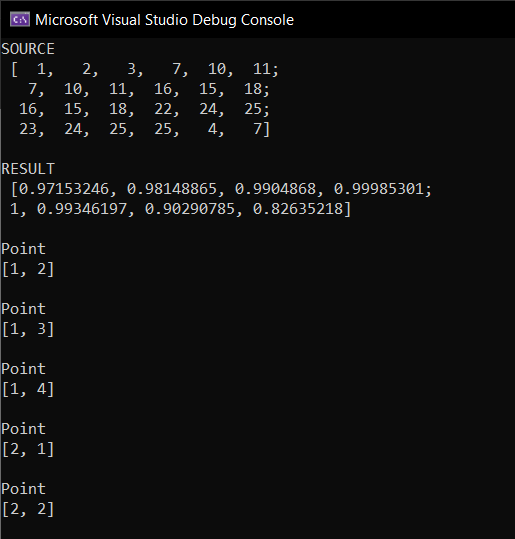
Hình 5: P5(2,2) giá trị P = 0.9935

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 7 | 10 | 11 |
| 7 | 10 | 11 | 16 | 15 | 18 |
| 16 | 15 | 18 | 22 | 24 | 25 |
| 23 | 24 | 25 | 25 | 4 | 7 |

b) Sử dụng hàm template matching của opencv để so sánh với kết quả câu a và b

|  |
| --- |
| // Khai báo  #include "opencv2/imgproc/imgproc.hpp"  #include "opencv2/highgui/highgui.hpp"  #include <stdlib.h>  #include <stdio.h>  #include <iostream>  using namespace cv;  using namespace std;  int main(int argc, char\*\* argv)  {  // Khai bao va xuat ma tran ban dau 4x6  Mat source = (Mat\_<uint8\_t>(4, 6) << 1, 2, 3, 7, 10, 11, 7, 10, 11, 16, 15, 18, 16, 15, 18, 22, 24, 25, 23, 24, 25, 25, 4, 7);  cout << "SOURCE" << endl << " " << source << endl << endl;  //Khai báo Template = kernel 3x3 và ma tran dst  Mat kernel = (Mat\_<uint8\_t>(3, 3) << 7, 10, 11, 16, 15, 18, 23, 24, 25);  Mat dst;  //Su dung ham MatchTemplate va hien thi ket qua ra ma tran “RESULT” 2x4  matchTemplate(source, kernel, dst, TM\_CCORR\_NORMED);  cout << "RESULT" << endl << " " << dst << endl << endl;  //Hien thi cac toa do > 0.98 trong ma tran "RESULT" 2x4  for (int i = 0; i < dst.rows; i++)  for (int j = 0; j < dst.cols; j++)  {  if (dst.at<float>(i, j) > 0.98)  cout << "Point" << endl << Point(i + 1, j + 1) << endl << endl;  }  waitKey(0);  return 0;  } |

**Kết quả:**



= > Kết quả từ Code ( câu b ) so với tính tay ( câu a ) là giống nhau.

**Bài 2:** Cho hình A và Kernel B và C với origin là vị trí số có gạch dưới như sau

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

Hình A:

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

Vị trí anchor point:

Do STT của nhóm là 7 nên vị trí anchor point là (0; -2)

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

Kernel B:

Flip kernel B thu được kernel B’:

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

1. **open by hand sử dụng Kernel B**

Ta có:

* Erode ma trận A với kernel B:
* Do vị trí anchor point của kernel B nên 2 dòng đầu trả về 0, chỉ cần xét erosion từ hàng thứ 3:

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

* Tại vị trí đầu tiên, các vị trí có giá trị 1 không khớp nên trả về 0:

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

* Xét vị trí tiếp theo, các vị trí có giá trị 1 không khớp nên trả về 0:

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

* Xét vị trí như hình bên dưới, các vị trí có giá trị 1 khớp nên trả về 1:

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

* Tương tự cho các ô còn lại ta được:

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

* Dilate ma trận đó với kernel B’:
* Xét tại vị trí đầu tiên, do không giống với kernel B’ nên trả về 0:

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

* Xét tại vị trí tiếp theo, do không giống với kernel B’ nên trả về 0:

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

* Xét vị trí như hình bên dưới, có vị trí có giá trị 1 khớp nên trả về 1:

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

* Tương tự cho các ô còn lại ta được:

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

1. **using open morphologyex function to compare sử dụng Kernel B**

|  |
| --- |
| #include "opencv2/highgui.hpp"  #include "opencv2/imgcodecs.hpp"  #include "opencv2/imgproc.hpp"  #include <iostream>  using namespace cv;  using namespace std;  int main(int argc, char\*\* argv)  {  //Nhap va in ma tran ban dau  Mat src = (Mat\_<float>(6,8) << 0, 0, 0, 0, 0, 0, 0, 0,  0, 0, 0, 1, 1, 0, 0, 0,  0, 1, 1, 1, 1, 1, 1, 0,  0, 0, 1, 1, 1, 0, 1, 0,  0, 1, 1, 1, 1, 1, 1, 0,  0, 0, 0, 0, 0, 0, 0, 0);  //Khoi tao kernel B  int mor\_size = 1;  Mat kernel = getStructuringElement(MORPH\_CROSS, Size(2 \* mor\_size + 1, 2 \* mor\_size + 1), Point(1, 1));  kernel.at<uchar>(1, 0) = 0;  Mat output1, output2;  //Tao ma tran flip cua B  Mat kernel\_flipped;  flip(kernel, kernel\_flipped, -1);  //Ham morphologyEx  //Tien hanh erode  morphologyEx(src, output1, MORPH\_ERODE, kernel, Point(0, 2), 1, BORDER\_CONSTANT, 0);  //Tien hanh dilate  morphologyEx(output1, output2, MORPH\_DILATE, kernel\_flipped, Point(2, 0), 1, BORDER\_CONSTANT, 0);  //In ket qua  cout << "Matrix after erosing" << endl;  cout << output1 << endl;  cout << "Opening using morphologyEx function" << endl;  cout << output2 << endl;  return 0;  } |
|  |

1. **close by hand sử dụng Kernel B**

Ta có:

* Dilate ma trận A với kernel B’:
* Xét tại vị trí như hình bên dưới, có vị trí có giá trị 1 khớp nên trả về 1:

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

* Xét tại vị trí tiếp theo, có vị trí có giá trị 1 khớp nên trả về 1:

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

* Xét tại vị trí tiếp theo, có vị trí có giá trị 1 khớp nên trả về 1:

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

* Tương tự cho các ô còn lại ta được:

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

* Erode ma trận đó với kernel B:
* Xét tại vị trí như hình bên dưới, các vị trí có giá trị 1 không khớp nên trả về 0:

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

* Xét tại vị trí tiếp theo, các vị trí có giá trị 1 khớp nên trả về 1:

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

* Xét tại vị trí tiếp theo, các vị trí có giá trị 1 khớp nên trả về 1:

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

* Tương tự cho các ô còn lại ta được:

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

1. **using close morphologyex function to compare sử dụng Kernel B**

|  |
| --- |
| #include "opencv2/highgui.hpp"  #include "opencv2/imgcodecs.hpp"  #include "opencv2/imgproc.hpp"  #include <iostream>  using namespace cv;  using namespace std;  int main(int argc, char\*\* argv)  {  //Nhap va in ma tran ban dau  Mat src = (Mat\_<float>(6,8) << 0, 0, 0, 0, 0, 0, 0, 0,  0, 0, 0, 1, 1, 0, 0, 0,  0, 1, 1, 1, 1, 1, 1, 0,  0, 0, 1, 1, 1, 0, 1, 0,  0, 1, 1, 1, 1, 1, 1, 0,  0, 0, 0, 0, 0, 0, 0, 0);  //Khoi tao kernel B  int mor\_size = 1;  Mat kernel = getStructuringElement(MORPH\_CROSS, Size(2 \* mor\_size + 1, 2 \* mor\_size + 1), Point(1, 1));  kernel.at<uchar>(1, 0) = 0;  Mat output1, output2;  //Tao ma tran flip cua B  Mat kernel\_flipped;  flip(kernel, kernel\_flipped, -1);  //Ham morphologyEx  //Tien hanh dilate  morphologyEx(src, output1, MORPH\_DILATE, kernel\_flipped, Point(2, 0), 1, BORDER\_CONSTANT, 0);  //Tien hanh erode  morphologyEx(output1, output2, MORPH\_ERODE, kernel, Point(0, 2), 1, BORDER\_CONSTANT, 0);  //In ket qua  cout << "Matrix after dilating" << endl;  cout << output1 << endl;  cout << "Closing using morphologyEx function" << endl;  cout << output2 << endl;  return 0;  } |
|  |

**Câu 3:** Cho hình sau.

A picture containing icon

Description automatically generated

1. Hiển thị chỉ đường tròn
2. Hiển thị chỉ hình vuông
3. Hiển thị chỉ đường thẳng ngang và đứng
4. Hiển thị đường thẳng đứng và đường xéo (450)

Phần lập trình

#include <opencv2/opencv.hpp>

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <stdint.h>

using namespace cv;

using namespace std;

//Tao ma tran Kernel duong cheo 45 do

Mat createKernel(int rows, int cols, int weight)

{

Mat kernel;

kernel = Mat::zeros(rows, cols, CV\_8UC1);

int i = 0, j = 0;

int x = 0;

for (i = 0; i < rows; i++)

{

for (j = 0; j < weight; j++)

{

x = cols - 1 - i - j;

if (x < 0) x = 0;

kernel.at<uchar>(i, x) = 1;

}

}

return kernel;

}

void main()

{

Mat img = imread("C:/Users/dmin/Desktop/Mor1.png");

Mat gray;

cvtColor(img, gray, CV\_8UC1);

imshow("Gray", gray);

//ab.Hien thi chi duong tron va chi hinh vuong

Mat dst, dst\_tron, dst\_vuong;

//Loc hinh tron va vuong

Mat kernel1;

kernel1 = getStructuringElement(MORPH\_ELLIPSE, Size(11, 11), Point(-1, -1));

morphologyEx(gray, dst, MORPH\_OPEN, kernel1, Point(-1, -1));

//Loc hinh vuong

Mat kernel2;

kernel2 = getStructuringElement(MORPH\_RECT, Size(21, 21), Point(-1, -1));

morphologyEx(gray, dst\_vuong, MORPH\_OPEN, kernel2, Point(-1, -1));

//Loc hinh tron

dst\_tron = dst - dst\_vuong;

Mat kernel3;

kernel3 = getStructuringElement(MORPH\_ELLIPSE, Size(5, 5), Point(-1, -1));

morphologyEx(dst\_tron, dst\_tron, MORPH\_OPEN, kernel3, Point(-1, -1));

imshow("chi duong tron", dst\_tron); //Anh hinh tron

imshow("tron+vuong", dst);

imshow("chi hinh vuong", dst\_vuong); //Anh hinh vuong

//c.Hien thi canh ngang va canh dung

//Loc canh ngang

Mat dst\_ngang, dst\_dung;

Mat Line = gray - dst;

Mat kernel4 = getStructuringElement(MORPH\_RECT, Size(9, 1), Point(-1, -1));

morphologyEx(Line, dst\_ngang, MORPH\_OPEN, kernel4, Point(-1, -1), 2, 0);

//Loc canh dung

Mat kernel5 = getStructuringElement(MORPH\_RECT, Size(1, 9), Point(-1, -1));

morphologyEx(Line, dst\_dung, MORPH\_OPEN, kernel5, Point(-1, -1), 2, 0);

Mat dst\_ngang\_dung = dst\_ngang + dst\_dung;

imshow("canh ngang va canh dung", dst\_ngang\_dung); //Anh canh ngang va canh dung

//d.Hien thi canh dung va canh cheo

//Loc duong cheo

Mat dst\_cheo; //Duong cheo 45

Mat kernel6 = createKernel(7, 7, 2);

morphologyEx(Line, dst\_cheo, MORPH\_OPEN, kernel6, Point(-1, -1), 2, 0, BORDER\_CONSTANT);

Mat dst\_cheo2; //Duong cheo 135

Mat kernel7;

flip(kernel6, kernel7, 1);

//cout << kernel7 << endl << kernel6;

morphologyEx(Line, dst\_cheo2, MORPH\_OPEN, kernel7, Point(-1, -1), 2, 0, BORDER\_CONSTANT);

dst\_cheo = dst\_cheo + dst\_cheo2;

Mat dst\_cheo\_dung;

dst\_cheo\_dung = dst\_cheo + dst\_dung;

imshow("", dst\_cheo\_dung);

waitKey(0);

}

Kết quả hiển thị:

a. Hiển thị đường tròn

A screenshot of a game

Description automatically generated with low confidence

b. Hiển thị hình vuông

A picture containing calendar

Description automatically generated

c. Hiển thị cạnh ngang và cạnh đứng

Text

Description automatically generated

d.Hiển thị cạnh đứng và cạnh chéo

**Text

Description automatically generated**

**Câu 4:**

Cho hình sau: A

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

Kernel:

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

a. Điền đầy hình sử dụng fill hole bằng tay, show tất cả các interaction.

Xác định ma trận:

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

Công thức tính của Hole Filling:

Tính toán

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **1** | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 |  | 1 | 0 | | 0 | 1 |  |  | 1 | 0 | | 0 | 1 | 1 |  | 1 | 0 | | 0 | 1 |  |  | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 |  | 1 | 1 | 0 | | 0 | 1 | 1 |  | 1 | 0 | | 0 | 1 |  |  | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 |  | 1 | 1 | 0 | | 0 | 1 | 1 |  | 1 | 0 | | 0 | 1 |  |  | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | |
| **2** | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 |  | 1 | 1 | 0 | | 0 | 1 | 1 |  | 1 | 0 | | 0 | 1 |  |  | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 |  |  | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 |  |  | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | |
| **3** | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 |  |  | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 |  | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 |  | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | |
|  |  |  |  |
| **4** | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 |  | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | |
|  |  |  |  |