**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**THÀNH PHỐ HỒ CHÍ MINH**



**NHẬP MÔN THỊ GIÁC MÁY TÍNH**

**Bài tập Spatial Operator**

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:**

**\*STT = 7**

**\*Anchor Point : “7”**

Ma trận đầu vào (F):

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 9 |
| 8 | 7 | 10 | 11 |
| 15 | 16 | 15 | 18 |
| 7 | 23 | 24 | 25 |

Kernel (H):

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

1. Tính bằng tay Cross – Correlation:

Anchor Point : “7”

Thêm biên “0” tương ứng :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 2 | 3 | 9 | 0 | 0 |
| 8 | 7 | 10 | 11 | 0 | 0 |
| 15 | 16 | 15 | 18 | 0 | 0 |
| 7 | 23 | 24 | 25 | 0 | 0 |

Ta có:

Áp dụng công thức, ta tính được các phần tử của ma trận:

…

Tương tự , ta có kết quả:

Ma trận sau khi lọc (G):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 50 | 119 | 93 | 63 | 0 | 0 |
| 234 | 305 | 215 | 113 | 0 | 0 |
| 509 | 573 | 365 | 179 | 0 | 0 |
| 731 | 885 | 550 | 258 | 0 | 0 |

\*Lập trình tính Cross-Correlation sử dụng hàm filter 2D

|  |
| --- |
| //Khai bao thu vien  #include <opencv2/core.hpp>  #include <opencv2/video/video.hpp>  #include <opencv2/imgproc.hpp>  #include <opencv2/highgui.hpp>  #include <opencv2/imgcodecs.hpp>  #include <iostream>  using namespace std;  using namespace cv;  int main()  {  //Khai bao ma tran dau vao  float source[4][4] = {{1 ,2 ,3 ,9},{8 ,7 ,10,11},{15,16,15,18},{7 ,23,24,25}};  Mat source\_img(4, 4, CV\_32F, source);  //Hien thi ma tran dau vao  cout << "SOUCRE:" << endl << source\_img << endl << endl;  //Khai bao ma tran KERNEL  float kernel\_data[3][3] = {{1,2,3},{4,5,6},{7,8,9} };  Mat mask\_cross(3, 3, CV\_32F, kernel\_data);  //Hien thi ma tran KERNEL  cout << "KERNEL" << endl << mask\_cross << endl << endl;  Mat dst;  //Su dung ham filter2D de thuc hien Cross-Correclation, anchor Point là vị trí số 7  filter2D(source\_img, dst, source\_img.depth(), mask\_cross, Point(0,2), 0, 0);  //Hien thi ma tran ket qua  cout << "CROSS-CORRELATION:" << endl << dst << endl << endl;  waitKey(0);  return 0;  } |

Kết quả:



**Kết quả từ Code giống với kết quả tính tay**

1. Tính bằng tay Convolution:

Tương tự như Cross – Correlation, chỉ thay đổi ma trận kernel (H) bằng việc đảo ngược theo hàng và cột sẽ được:

Ta có Kernel:

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

Flip Kernel ta được :

\_Bước 1: đổi vị trí hàng 1 và hàng 3

\_Bước 2: đổi vị trí cột 1 và cột 3

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

Anchor Point là “vị trí” của số “7” ban đầu.

Thêm biên “0” tương ứng (F):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 2 | 3 | 9 | 0 | 0 |
| 8 | 7 | 10 | 11 | 0 | 0 |
| 15 | 16 | 15 | 18 | 0 | 0 |
| 7 | 23 | 24 | 25 | 0 | 0 |

Ta có:

:

Áp dụng công thức, ta tính được các phần tử của ma trận:

…

Tương tự , ta có kết quả:

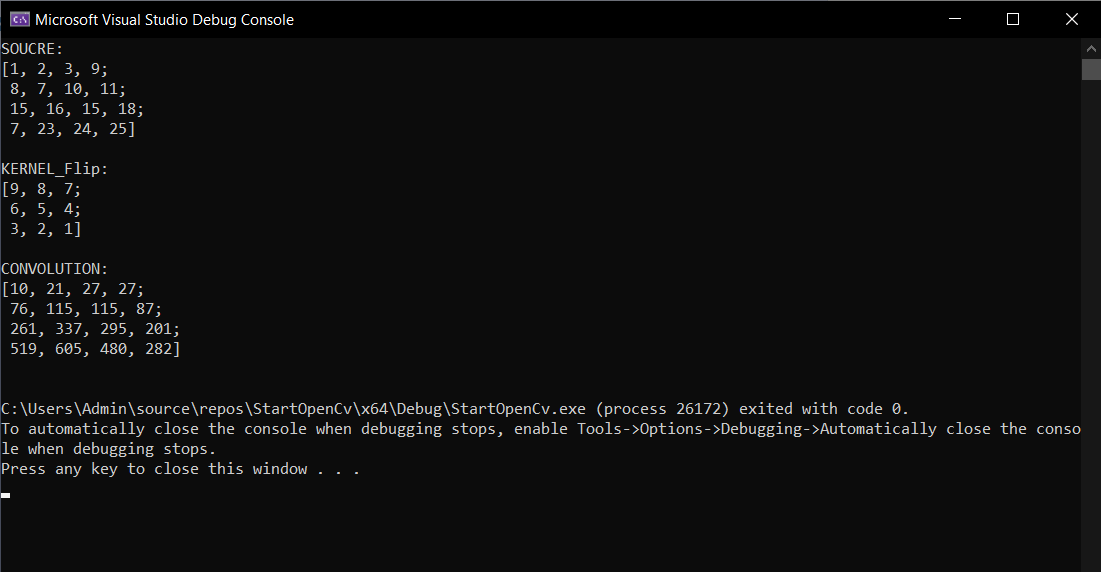
Ma trận sau khi lọc (G):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 21 | 27 | 27 | 0 | 0 |
| 76 | 115 | 115 | 87 | 0 | 0 |
| 261 | 337 | 295 | 201 | 0 | 0 |
| 519 | 605 | 480 | 282 | 0 | 0 |

\*Lập trình tính Convolution sử dụng hàm filter2D:

|  |
| --- |
| #include <opencv2/core.hpp>  #include <opencv2/video/video.hpp>  #include <opencv2/imgproc.hpp>  #include <opencv2/highgui.hpp>  #include <opencv2/imgcodecs.hpp>  #include <iostream>  using namespace std;  using namespace cv;  int main()  {  //Khai bao ma tran dau vao  float source[4][4] = { {1,2,3,9},{8,7,10,11},{15,16,15,18},{7,23,24,25} };  Mat source\_img(4, 4, CV\_32F, source);  //Hien thi ma tran dau vao  cout << "SOUCRE:" << endl << source\_img << endl << endl;  //Khai bao ma tran KERNEL  float kernel\_data[3][3] = { {1,2,3},{4,5,6},{7,8,9}};  Mat mask\_cross(3, 3, CV\_32F, kernel\_data);  Mat mask\_conv;  //FLip ma trận mặt nạ KERNEL  flip(mask\_cross, mask\_conv, -1);  // Hien thi ma trận mặt nạ KERNEL\_Flip su dung de thuc hien Convolution  cout << "KERNEL\_Flip:" << endl << mask\_conv << endl << endl;  Mat dst;  //Su dung ham filter2D de thuc hien Convolution, anchor Point là vị trí số 7  filter2D(source\_img, dst, -1, mask\_conv, Point(0, 2), 0, 0);  cout << "CONVOLUTION:" << endl << dst << endl << endl;  waitKey(0);  return 0;  } |

**Kết quả:**



**Kết quả từ code giống so với tính tay.**

**Bài 2:**

**Cho hình có ma trận sau:**

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **2** | **3** | **STT+2** |
| **8** | **STT** | **10** | **11** |
| **15** | **16** | **STT+8** | **18** |
| **STT** | **23** | **24** | **25** |

1. Tính bằng tay lọc trung bình của hình trên với kernel 3x3
2. Lập trình tính lọc trung bình ma trận trên sử dụng hàm filter2D, so sánh với kết quả a, b

Bài làm:

1. Ta có ma trận H ban đầu (STT = 7):

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 9 |
| 8 | 7 | 10 | 11 |
| 15 | 16 | 15 | 18 |
| 7 | 23 | 24 | 25 |

Thêm padding 0 tương ứng với anchor Point của bài 1 ( vị trí số 7 )

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 2 | 3 | 9 | 0 | 0 |
| 8 | 7 | 10 | 11 | 0 | 0 |
| 15 | 16 | 15 | 18 | 0 | 0 |
| 7 | 23 | 24 | 25 | 0 | 0 |

- Ta có ma trận Kernel 3x3:

|  |  |  |
| --- | --- | --- |
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |

\* vị trí anchor Point lấy theo câu 1

Tacó:

Áp dụng công thức, ta tính được các phần tử của ma trận:

…

Tính cho các điểm còn lại.

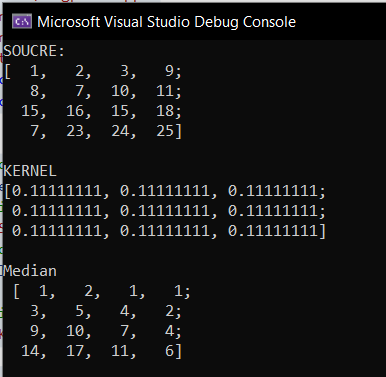
-Suy ra ta có ma trận sau khi lọc trung bình M

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 2 | 1 | 1 | 0 | 0 |
| 3 | 5 | 4 | 2 | 0 | 0 |
| 9 | 10 | 7 | 4 | 0 | 0 |
| 14 | 17 | 11 | 6 | 0 | 0 |

1. Lập trình tính lọc trung bình ma trận trên sử dụng hàm filter2D, so sánh với kết quả a, b

|  |
| --- |
| //Khai bao thu vien  #include <opencv2/core.hpp>  #include <opencv2/video/video.hpp>  #include <opencv2/imgproc.hpp>  #include <opencv2/highgui.hpp>  #include <opencv2/imgcodecs.hpp>  #include <iostream>  using namespace std;  using namespace cv;  int main()  {  //Khai bao ma tran dau vao  Mat source\_img = (Mat\_<uint8\_t>(4, 4) << 1, 2, 3, 9, 8, 7, 10, 11, 15, 16, 15, 18, 7, 23, 24, 25);  //Hien thi ma tran dau vao  cout << "SOUCRE:" << endl << source\_img << endl << endl;  //Khai bao ma tran KERNEL  Mat kernel3x3;  kernel3x3 = Mat::ones(3, 3, CV\_32F) / (float)(3 \* 3);  //Hien thi ma tran KERNEL  cout << "KERNEL" << endl << kernel3x3 << endl << endl;  Mat dst;  //Su dung ham filter2D de thuc hien Loc trung binh, anchor Point là vị trí số 7  filter2D(source\_img, dst, source\_img.depth(), kernel3x3, Point(0, 2), 0, 0);  //Hien thi ma tran loc trung binh  cout << "Median " << endl << " " << dst << endl << endl;  waitKey(0);  return 0;  } |

**Kết quả:**



**Vậy kết quả từ Code và tính tay giống nhau**

**Câu 3:** Cho hình có ma trận sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT+1** | **2** | **3** | **4** | **5** |
| **8** | **STT+2** | **10** | **11** | **12** |
| **15** | **16** | **STT+3** | **18** | **19** |
| **22** | **23** | **24** | **STT+4** | **26** |
| **29** | **30** | **31** | **32** | **STT+5** |

**Với STT = 7 ta được:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **8** | **2** | **3** | **4** | **5** |
| **8** | **9** | **10** | **11** | **12** |
| **15** | **16** | **10** | **18** | **19** |
| **22** | **23** | **24** | **11** | **26** |
| **29** | **30** | **31** | **32** | **12** |

1. Tính bằng tay lọc trung vị của ma trận trên với kernel 3x3:

Ta có:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **8** | **2** | **3** | **4** | **5** |
| **8** | **9** | **10** | **11** | **12** |
| **15** | **16** | **10** | **18** | **19** |
| **22** | **23** | **24** | **11** | **26** |
| **29** | **30** | **31** | **32** | **12** |

Thêm Border vào ta được:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **8** | **8** | **2** | **3** | **4** | **5** | **5** |
| **8** | **8** | **2** | **3** | **4** | **5** | **5** |
| **8** | **8** | **9** | **10** | **11** | **12** | **12** |
| **15** | **15** | **16** | **10** | **18** | **19** | **19** |
| **22** | **22** | **23** | **24** | **11** | **26** | **26** |
| **29** | **29** | **30** | **31** | **32** | **12** | **12** |
| **29** | **29** | **30** | **31** | **32** | **12** | **12** |

Lọc lần lượt từng khoảng với Kernel 3x3:

**Lần 1:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **8** | **8** | **2** | **3** | **4** | **5** | **5** |
| **8** | **8** | **2** | **3** | **4** | **5** | **5** |
| **8** | **8** | **9** | **10** | **11** | **12** | **12** |
| **15** | **15** | **16** | **10** | **18** | **19** | **19** |
| **22** | **22** | **23** | **24** | **11** | **26** | **26** |
| **29** | **29** | **30** | **31** | **32** | **12** | **12** |
| **29** | **29** | **30** | **31** | **32** | **12** | **12** |

Ta có giá trị trung vị là: {2 2 8 8 **8** 8 8 8 9}→8

**Lần 2:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **8** | **8** | **2** | **3** | **4** | **5** | **5** |
| **8** | **8** | **2** | **3** | **4** | **5** | **5** |
| **8** | **8** | **9** | **10** | **11** | **12** | **12** |
| **15** | **15** | **16** | **10** | **18** | **19** | **19** |
| **22** | **22** | **23** | **24** | **11** | **26** | **26** |
| **29** | **29** | **30** | **31** | **32** | **12** | **12** |
| **29** | **29** | **30** | **31** | **32** | **12** | **12** |

Ta có giá trị trung vị là: {2 2 3 3 **8** 8 8 9 10}→8

**Lần 3:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **8** | **8** | **2** | **3** | **4** | **5** | **5** |
| **8** | **8** | **2** | **3** | **4** | **5** | **5** |
| **8** | **8** | **9** | **10** | **11** | **12** | **12** |
| **15** | **15** | **16** | **10** | **18** | **19** | **19** |
| **22** | **22** | **23** | **24** | **11** | **26** | **26** |
| **29** | **29** | **30** | **31** | **32** | **12** | **12** |
| **29** | **29** | **30** | **31** | **32** | **12** | **12** |

Ta có giá trị trung vị là: {2 2 3 3 **4** 4 9 10 11}→4

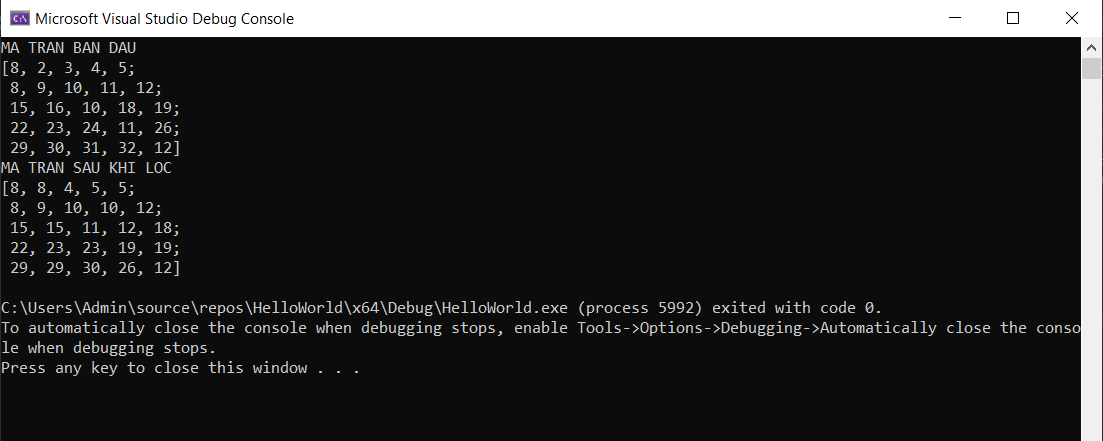
Tương tự cho phần còn lại, ta thu được ma trận sau khi lọc:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **8** | **8** | **4** | **5** | **5** |
| **8** | **9** | **10** | **10** | **12** |
| **15** | **15** | **11** | **12** | **18** |
| **22** | **23** | **23** | **19** | **19** |
| **29** | **29** | **30** | **26** | **12** |

1. Lập trình tính lọc trung vị ma trận trên sử dụng hàm Medianblur:

|  |
| --- |
| #include "opencv2/highgui.hpp"  #include "opencv2/imgcodecs.hpp"  #include "opencv2/imgproc.hpp"  #include <iostream>  using namespace cv;  using namespace std;  int main()  {  //KHOI TAO MA TRAN  Mat scr = (Mat\_<float>(5, 5) << 8, 2, 3, 4, 5,  8, 9, 10, 11, 12,  15, 16, 10, 18, 19,  22, 23, 24, 11, 26,  29, 30, 31, 32, 12);  cout << "MA TRAN BAN DAU\n" << scr << endl;  //MA TRAN OUTPUT  Mat dst;  //HAM LOC TRUNG VI – medianBlur voi kernel 3x3  medianBlur(scr, dst, 3);  //IN MA TRAN DA LOC RA MAN HINH  cout << "MA TRAN SAU KHI LOC\n" << dst << endl;  return 0;  } |
|  |

Kết quả:



**Câu 4:** STT = 7

Cho hình có ma trận:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 8 | 2 | 3 | 4 | 5 |
| 8 | 9 | 10 | 11 | 12 |
| 15 | 16 | 10 | 18 | 19 |
| 22 | 23 | 24 | 11 | 26 |
| 29 | 30 | 31 | 32 | 12 |

c) Tính bằng tay lọc Gaussian của ma trận trên với kernel 3x3, sigma = STT/10

d) Lập trình tính lọc Gaussian ma trận trên sử dụng hàm filter2D và Gassianblur so sánh với kết quả a

**Bài làm:**

c) sigma = 7/10 = 0.7

Ta có lưới tọa độ như sau:

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

Áp dụng công thức với σ = 0.7, ta được kernel 3x3 như sau

|  |  |  |
| --- | --- | --- |
| 0.042 | 0.117 | 0.042 |
| 0.117 | 0.325 | 0.117 |
| 0.042 | 0.117 | 0.042 |

Với các vị trí được tính như sau:

= 0.042

= 0.117

= 0.325

Tương tự cho các vị trí còn lại trong bảng.

Vì tổng các phần tử trong bảng kernel phải bằng 1 nên ta nhân mặt nạ trên với 1/ tổng phần tử của mặt nạ. Với 1/tổng các phần tử = 1/0.961 = 1.0406

Ta có kernel 3x3 mới như sau:

|  |  |  |
| --- | --- | --- |
| 0.043704 | 0.121748 | 0.043704 |
| 0.121748 | 0.338189 | 0.121748 |
| 0.043704 | 0.121748 | 0.043704 |

Viết lại ma trận đầu bài, thêm 1 hàng 0 và 1 cột 0:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 8 | 2 | 3 | 4 | 5 | 0 |
| 0 | 8 | 9 | 10 | 11 | 12 | 0 |
| 0 | 15 | 16 | 10 | 18 | 19 | 0 |
| 0 | 22 | 23 | 24 | 11 | 26 | 0 |
| 0 | 29 | 30 | 31 | 32 | 12 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Áp dụng kernel 3x3, ta được các phần tử có giá trị như sau:

a11 =0x0.042 + 0x0.117+ 0x0.042 + 0x0.117 + 8x0.325 + 2x0.117 + 0x0.042 + 8x0.117 + 9x0.042 = 4.148

Tương tự với các phần tử còn lại, ta được ma trận như sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4.316337 | 3.898023 | 3.836629 | 4.627471 | 4.119667 |
| 7.388137 | 9 | 9.147763 | 10.69407 | 9.280957 |
| 12.0718 | 15.14776 | 14.02081 | 15.44329 | 14.20499 |
| 17.6077 | 22.69407 | 21.44329 | 19.04162 | 16.09157 |
| 17.1436 | 22.26119 | 22.44017 | 19.58169 | 11.60042 |

d) Code

#include <opencv2/opencv.hpp>

#include <iostream>

#include<stdio.h>

#include<stdlib.h>

#include <stdint.h>

using namespace cv;

using namespace std;

int main(int argc, char\*\* argv)

{

// Read the matrix

Mat image = (Mat\_<double>(5, 5) << 8, 2, 3, 4, 5, 8, 9, 10, 11, 12, 15, 16, 10, 18, 19, 22, 23, 24, 11, 26, 29, 30, 31, 32, 12);

cout << "original image = " << endl << " " << image << endl << endl;

Mat image\_kernel;

GaussianBlur(image, image\_kernel, Size(3, 3), 0.7, 0, BORDER\_CONSTANT);

//Display Gaussian Blur matrix

cout << "Gaussian blur image = " << endl << " " << image\_kernel << endl << endl;

//Filter2D

//Calculate Kernel

int length = 3;

double sigma = 0.7;

Mat Kernel(length, length, DataType<double>::type);

double sumTotal = 0;

int kernelRadius = (length - 1) / 2;

double distance = 0;

double calculatedEuler = 1.0 / (2.0 \* CV\_PI \* sigma \* sigma);

for (int filterY = -kernelRadius; filterY <= kernelRadius; filterY++)

{

for (int filterX = -kernelRadius; filterX <= kernelRadius; filterX++)

{

distance = ((filterX \* filterX) + (filterY \* filterY)) / (2 \* (sigma \* sigma));

Kernel.at<double>(filterY + kernelRadius, filterX + kernelRadius) = calculatedEuler \* exp(-distance);

sumTotal += Kernel.at<double>(filterY + kernelRadius, filterX + kernelRadius);

}

}

for (int y = 0; y < length; y++)

{

for (int x = 0; x < length; x++)

{

Kernel.at<double>(y, x) = Kernel.at<double>(y, x) \* (1.0 / sumTotal);

}

}

//Display Filter2D matrix

Mat image\_after\_blur\_by\_filter2d;

filter2D(image, image\_after\_blur\_by\_filter2d, -1, Kernel, Point(-1, -1), 0, BORDER\_CONSTANT);

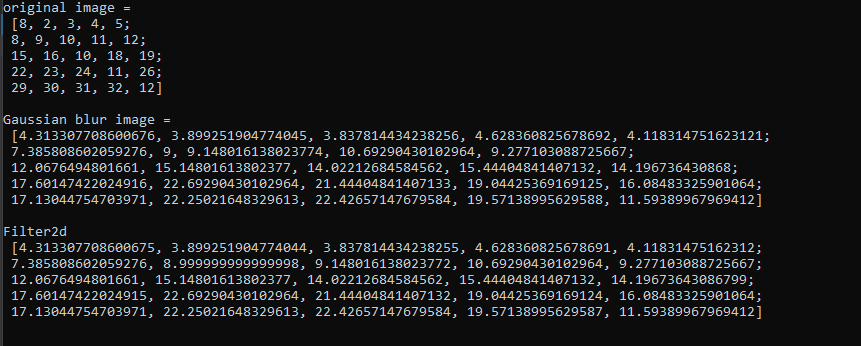
cout << "Filter2d " << endl << " " << image\_after\_blur\_by\_filter2d << endl << endl;

waitKey(0);

return 0;

}

Kết quả:



Câu 5:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 9 | 10 | 11 | 40 | 9 | 10 | 11 |
| 10 | 9 | 40 | 11 | 40 | 10 | 11 |
| 10 | 40 | 10 | 11 | 10 | 40 | 11 |
| 10 | 9 | 40 | 11 | 40 | 13 | 11 |
| 10 | 10 | 10 | 40 | 10 | 13 | 11 |
| 10 | 10 | 10 | 10 | 10 | 10 | 10 |

a. Tính bằng tay sobel theo x và y:

Thêm padding 0 và ma trận đề bài.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 9 | 10 | 11 | 40 | 9 | 10 | 11 | 0 |
| 0 | 10 | 9 | 40 | 11 | 40 | 10 | 11 | 0 |
| 0 | 10 | 40 | 10 | 11 | 10 | 40 | 11 | 0 |
| 0 | 10 | 9 | 40 | 11 | 40 | 13 | 11 | 0 |
| 0 | 10 | 10 | 10 | 40 | 10 | 13 | 11 | 0 |
| 0 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Tính Sobel theo x:**

Kernel theo phương x, với anchor point là điểm 0 ở giữa:

|  |  |  |
| --- | --- | --- |
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

Tính toán:

Đầu tiên ta để kernel lên trên khung đậm, sau đó tính toán,

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 9 | 10 | 40 | 9 |
| 0 | 10 | 9 | 11 | 40 |
| 0 | 10 | 40 | 11 | 40 |
| 0 | 10 | 9 | 11 | 40 |

Giá trị điểm anchor được xác định như sau:

Sau đó trượt kernel sang phải 1 ô:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 9 | 10 | 40 | 9 |
| 0 | 10 | 9 | 11 | 40 |
| 0 | 10 | 40 | 11 | 40 |
| 0 | 10 | 9 | 11 | 40 |

Giá trị tại điểm anchor được xác định như sau:

Tiếp tục dời kernel sang phải 1 ô:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 9 | 10 | 40 | 9 |
| 0 | 10 | 9 | 11 | 40 |
| 0 | 10 | 40 | 11 | 40 |
| 0 | 10 | 9 | 11 | 40 |

Giá trị tại điểm anchor được xác định như sau:

Tính tương tự cho các điểm còn lại, sau khi đi hết cột, dời kernel xuống 1 hàng rồi sau đó tiến hành tính toán tiếp. Sau khi tính toán cho các điểm còn lại, ta có kết quả sau:

Theo x:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 29 | 34 | 62 | -4 | -61 | -25 | -30 |
| 68 | 62 | 5 | -2 | -3 | -55 | -70 |
| 98 | 60 | -54 | 0 | 59 | -56 | -103 |
| 68 | 60 | 5 | 0 | 6 | -56 | -79 |
| 39 | 30 | 62 | 0 | -52 | -27 | -49 |
| 30 | 0 | 30 | 0 | -27 | 1 | -33 |

**Tính sobel theo phương y:**

|  |  |  |
| --- | --- | --- |
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

Tính tương tự như với theo phương x, kết quả có được như sau:

Theo y:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 29 | 68 | 100 | 102 | 101 | 71 | 32 |
| 32 | 60 | -1 | -58 | 3 | 61 | 30 |
| 0 | 0 | 0 | 0 | 3 | 6 | 3 |
| -30 | -60 | -1 | 58 | 2 | -54 | -27 |
| 1 | -28 | -60 | -62 | -64 | -37 | -5 |
| -30 | -40 | -70 | -100 | -73 | -47 | -35 |

b. Lập trình tính sobel theo x và y dùng hàm filter2D và hàm sobel.

Code:

|  |
| --- |
| #include <opencv2/opencv.hpp>  #include <iostream>  #include<stdio.h>  #include<stdlib.h>  #include <stdint.h>  using namespace cv;  using namespace std;  int main(int argv, char\*\* argc) {  float anh[42] = { 9, 10, 11, 40, 9, 10, 11,  10, 9, 40, 11, 40, 10, 11,  10, 40, 10, 11, 10, 40, 11,  10, 9, 40, 11, 40, 13, 11,  10, 10, 10, 40, 10, 13, 11,  10, 10, 10, 10, 10, 10, 10  };  float ker\_x[9] = { -1, 0, 1, -2, 0, 2, -1, 0, 1 };  float ker\_y[9] = { -1, -2, -1, 0, 0, 0, 1, 2, 1 };  Mat src\_anh = Mat(6, 7, CV\_32F, anh);  Mat kerx = Mat(3, 3, CV\_32F, ker\_x);  Mat kery = Mat(3, 3, CV\_32F, ker\_y);  Mat output\_X; Mat output\_Y;  Mat output\_x\_2D; Mat output\_y\_2D;  cv::filter2D(src\_anh, output\_x\_2D, CV\_32F, kerx, Point(-1, -1), 0, 0);  cv::filter2D(src\_anh, output\_y\_2D, CV\_32F, kery, Point(-1, -1), 0, 0);  cv::Sobel(src\_anh, output\_X, CV\_32F, 1, 0, 3, 1, 0, 0);  cv::Sobel(src\_anh, output\_Y, CV\_32F, 0, 1, 3, 1, 0, 0);  cout << "src = " << endl << " " << src\_anh << endl << endl;  cout << "loc\_X\_sobel = " << endl << " " << output\_X << endl << endl;  cout << "loc\_X\_2D = " << endl << " " << output\_x\_2D << endl << endl;  cout << "loc\_Y\_sobel = " << endl << " " << output\_Y << endl << endl;  cout << "loc\_Y\_2D = " << endl << " " << output\_y\_2D << endl << endl;  } |
|  |

Kết quả:

