高级图像处理与分析课程实验



高级图像处理与分析课程实验报告

学号 JG15225001

姓名 祁振东

日期２０１６年４月１０日

|  |  |
| --- | --- |
| **实验名称** | 空域滤波 |
| **实验内容** | 1、利用均值模板平滑灰度图像。 具体内容：利用OpenCV对图像像素进行操作，分别利用3\*3、5\*5和9\*9尺寸的均值模板平滑灰度图像  2、利用高斯模板平滑灰度图像。 具体内容：利用OpenCV对图像像素进行操作，分别利用3\*3、5\*5和9\*9尺寸的高斯模板平滑灰度图像  3、利用Laplacian、Robert、Sobel模板锐化灰度图像。 具体内容：利用OpenCV对图像像素进行操作，分别利用Laplacian、Robert、Sobel模板锐化灰度图像  4、利用高提升滤波算法增强灰度图像。 具体内容：利用OpenCV对图像像素进行操作，设计高提升滤波算法增强图像  5、利用均值模板平滑彩色图像。 具体内容：利用OpenCV分别对图像像素的RGB三个通道进行操作，利用3\*3、5\*5和9\*9尺寸的均值模板平滑彩色图像  6、利用高斯模板平滑彩色图像。 具体内容：利用OpenCV分别对图像像素的RGB三个通道进行操作，分别利用3\*3、5\*5和9\*9尺寸的高斯模板平滑彩色图像  7、利用Laplacian、Robert、Sobel模板锐化灰度图像。 具体内容：利用OpenCV分别对图像像素的RGB三个通道进行操作，分别利用Laplacian、Robert、Sobel模板锐化彩色图像 |
| **实验完成情况** | [均匀模板（彩色和灰色通用）](#a)  [高斯平滑（彩色和灰色通用）](#b)  [Sobel，Robert，Laplacian（彩色和灰色通用）](#c) |
| （包括完成的 |
| 实验内容及 |
| 每个实验的 |
| 完成程度。 |
| 注意要贴出 |
| 每个实验的 |
| 核心代码） |
| **实验中的问题** | **多次尝试得出以下参数，其效果比书上的还要好**  拉普拉斯变换公式：X＝(拉普拉斯\*0.5+X)(标定)  Robert变换x=变换值（标定）  Sobel变换x=x+变换值\*0.25（标定） |
| （包括在实验 |
| 中遇到的问 |
| 题，以及解 |
| 决问题的方 |
| 法） |
| **实验结果** | 实验中我以彩色图像为例进行变换，灰色图像同理。代码也同样适用灰色图像。 |
| （实验完成后 |
| 的源码和打 |
| 包文件的说 |
| 明） |

**一、均值模板（彩色和灰色通用）**

|  |
| --- |
| #include <opencv\highgui.h>  #include <cv.h>  #include <vector>  #define N 9  #define path "..\\..\\pic\\mg.jpg"  using namespace cv;  using namespace std;  double arr[N][N] = { { 0 } };  void inital(double arr[][N]);  void JZ(Mat src);  uchar compute(int bb[][N]);  int main()  {  inital(arr);  Mat src = imread(path);  //imshow("aaa",src);  JZ(src);  waitKey(0);  return 0;  }  void JZ(Mat src)  {  int bb[N][N] = { { 0 } };  int i = src.rows, j = src.cols;  int w = 0, h = 0;  cout << i << "\*" << j;  for (i = 1; i < src.rows - 1 - N; i++)  {  for (j = 1; j < src.cols - 1 - N; j++)  {  //第一个通道  for (w = 0; w < N; w++)  {  for (h = 0; h < N; h++)  {  bb[w][h] = (int)src.at<Vec3b>(i - 1 + w, j - 1 + h)[0];  }  }  (uchar)src.at<Vec3b>(i, j)[0] = compute(bb);  //第二个通道  for (w = 0; w < N; w++)  {  for (h = 0; h < N; h++)  {  bb[w][h] = (int)src.at<Vec3b>(i - 1 + w, j - 1 + h)[1];  }  }  (uchar)src.at<Vec3b>(i, j)[1] = compute(bb);  //第三个通道  for (w = 0; w < N; w++)  {  for (h = 0; h < N; h++)  {  bb[w][h] = (int)src.at<Vec3b>(i - 1 + w, j - 1 + h)[2];  }  }  (uchar)src.at<Vec3b>(i, j)[2] = compute(bb);  }  }  imshow("aa", src);  }  void inital(double arr[][N])  {  for (int i = 0; i < N; i++)  {  for (int j = 0; j < N; j++)  {  arr[i][j] = (double)1 / (N\*N);  }  }  }  uchar compute(int bb[][N])  {  int temp = 0;  for (int i = 0; i < N; i++)  {  for (int j = 0; j < N; j++)  {  temp = arr[i][j] \* bb[i][j] + temp;  }  }  return temp;  } |

**二、高斯模板**

|  |
| --- |
| #include <opencv\highgui.h>  #include <cv.h>  #include <vector>  #define N 5  #define path "..\\..\\pic\\mg.jpg"  using namespace cv;  using namespace std;  double arr[N][N] = { { 0 } };  void inital33(double arr[][N]);  void inital55(double arr[][N]);  void JZ(Mat src);  int compute(int bb[][N]);  int main()  {  inital55(arr);  Mat src = imread(path);  //imshow("aaa",src);  JZ(src);  waitKey(0);  return 0;  }  void JZ(Mat src)  {  int bb[N][N] = { { 0 } };  int i = src.rows, j = src.cols;  int w = 0, h = 0;  cout << i << "\*" << j;  for (i = N / 2; i < src.rows - 1 - N/2; i++)  {  for (j = N / 2; j < src.cols - 1 - N/2; j++)  {  //第一个通道  for (w = 0; w < N; w++)  {  for (h = 0; h < N; h++)  {  bb[w][h] = (int)src.at<Vec3b>(i - N / 2 + w, j - N / 2 + h)[0];  }  }  src.at<Vec3b>(i, j)[0] = compute(bb);  //第二个通道  for (w = 0; w < N; w++)  {  for (h = 0; h < N; h++)  {  bb[w][h] = (int)src.at<Vec3b>(i - N / 2 + w, j - N / 2 + h)[1];  }  }  src.at<Vec3b>(i, j)[1] = compute(bb);  //第三个通道  for (w = 0; w < N; w++)  {  for (h = 0; h < N; h++)  {  bb[w][h] = (int)src.at<Vec3b>(i - N / 2 + w, j - N / 2 + h)[2];  }  }  src.at<Vec3b>(i, j)[2] = compute(bb);  }  }  imshow("高斯滤波", src);  }  void inital33(double arr[][N])  {  arr[0][0] = 1;  arr[0][1] = 2;  arr[0][3] = 1;  arr[1][0] = 2;  arr[1][1] = 4;  arr[1][2] = 2;  arr[2][0] = 1;  arr[2][1] = 2;  arr[2][2] = 1;  }  void inital55(double arr[][N])  {  arr[0][0] = 1;  arr[0][1] = 4;  arr[0][2] = 7;  arr[0][3] = 4;  arr[0][4] = 1;  arr[1][0] = 4;  arr[1][1] = 16;  arr[1][2] = 26;  arr[1][3] = 16;  arr[1][4] = 4;  arr[2][0] = 7;  arr[2][1] = 26;  arr[2][2] = 41;  arr[2][3] = 26;  arr[2][4] = 7;  arr[3][0] = 4;  arr[3][1] = 16;  arr[3][2] = 26;  arr[3][3] = 16;  arr[3][4] = 4;  arr[4][0] = 1;  arr[4][1] = 4;  arr[4][2] = 7;  arr[4][3] = 4;  arr[4][4] = 1;  }  int compute(int bb[][N])  {  double temp = 0;  for (int i = 0; i < N; i++)  {  for (int j = 0; j < N; j++)  {  temp = arr[i][j] \* bb[i][j] + temp;  }  }  return temp/(273);  } |

**三、Sobel，Robert，Laplacian（彩色和灰色通用）**

|  |
| --- |
| #include <opencv\highgui.h>  #include <cv.h>  #include <vector>  #define N 3  #define path "..\\..\\pic\\yq.tif"  using namespace cv;  using namespace std;  double arr[N][N] = { { 0 } };  static double max1 = 0;  static double min1 = 0;  void Laplacian();  void Robert();  void Sobel();  Mat LaplacianBH();  Mat RobertBH();  Mat SobelBH();  double compute(int bb[][N]);  void print();  int main()  {  //Sobel，Robert，Laplacian  Mat src = imread(path);  imshow("原始图像", src);  Laplacian();  cout << "Laplacian" << endl;  print();  imshow("Laplacian", LaplacianBH());  Robert();  cout << "Robert" << endl;  print();  imshow("Robert", RobertBH());  Sobel();  cout << "Sobel" << endl;  print();  imshow("Sobel", SobelBH());  //cvtColor(src, src, CV\_BGR2GRAY);//变换成灰度图像  //cout <<"min1="<< min1 << "\nmax1="<<max1 << endl;  waitKey(0);  return 0;  }  Mat LaplacianBH()  {  Mat src = imread(path);  int bb[N][N] = { { 0 } };  int temp = 0;  int i = src.rows, j = src.cols;  int w = 0, h = 0;  cout << i << "\*" << j;  for (i = 1; i < src.rows - 1 - N; i++)  {  for (j = 1; j < src.cols - 1 - N; j++)  {  //第一个通道  for (w = 0; w < N; w++)  {  for (h = 0; h < N; h++)  {  bb[w][h] = (int)src.at<Vec3b>(i - 1 + w, j - 1 + h)[0] ;  }  }  temp = (int)src.at<Vec3b>(i , j )[0] - compute(bb)\* 0.5;  temp=temp>255 ? 255 : temp;  temp = temp<0 ? 0 : temp;  src.at<Vec3b>(i, j)[0] = (int)temp;  //第二个通道  for (w = 0; w < N; w++)  {  for (h = 0; h < N; h++)  {  bb[w][h] = (int)src.at<Vec3b>(i - 1 + w, j - 1 + h)[1];  }  }  temp = (int)src.at<Vec3b>(i, j)[1] - compute(bb) \* 0.5;  temp = temp>255 ? 255 : temp;  temp = temp<0 ? 0 : temp;  src.at<Vec3b>(i, j)[1] = (int)temp;  //第三个通道  for (w = 0; w < N; w++)  {  for (h = 0; h < N; h++)  {  bb[w][h] = (int)src.at<Vec3b>(i - 1 + w, j - 1 + h)[2];  }  }  temp = (int)src.at<Vec3b>(i, j)[2] - compute(bb) \* 0.5;  temp = temp>255 ? 255 : temp;  temp = temp<0 ? 0 : temp;  src.at<Vec3b>(i, j)[2] = (int)temp;  }  }  return src;  }  Mat RobertBH()  {  Mat src = imread(path);  int bb[N][N] = { { 0 } };  int temp = 0;  int i = src.rows, j = src.cols;  int w = 0, h = 0;  cout << i << "\*" << j;  for (i = 1; i < src.rows - 1 - N; i++)  {  for (j = 1; j < src.cols - 1 - N; j++)  {  //第一个通道  for (w = 0; w < N; w++)  {  for (h = 0; h < N; h++)  {  bb[w][h] = (int)src.at<Vec3b>(i - 1 + w, j - 1 + h)[0];  }  }  temp = (int)src.at<Vec3b>(i, j)[0] + compute(bb);  temp = temp>255 ? 255 : temp;  temp = temp<0 ? 0 : temp;  src.at<Vec3b>(i, j)[0] = (int)temp;  //第二个通道  for (w = 0; w < N; w++)  {  for (h = 0; h < N; h++)  {  bb[w][h] = (int)src.at<Vec3b>(i - 1 + w, j - 1 + h)[1];  }  }  temp = (int)src.at<Vec3b>(i, j)[1] + compute(bb);  temp = temp>255 ? 255 : temp;  temp = temp<0 ? 0 : temp;  src.at<Vec3b>(i, j)[1] = (int)temp;  //第三个通道  for (w = 0; w < N; w++)  {  for (h = 0; h < N; h++)  {  bb[w][h] = (int)src.at<Vec3b>(i - 1 + w, j - 1 + h)[2];  }  }  temp = (int)src.at<Vec3b>(i, j)[2] + compute(bb);  temp = temp>255 ? 255 : temp;  temp = temp<0 ? 0 : temp;  src.at<Vec3b>(i, j)[2] = (int)temp;  }  }  return src;  }  Mat SobelBH()  {  Mat src = imread(path);  int bb[N][N] = { { 0 } };  int comp;  int temp = 0;  int i = src.rows, j = src.cols;  int w = 0, h = 0;  cout << i << "\*" << j;  for (i = N / 2; i < src.rows - 1 - N; i++)  {  for (j = N / 2; j < src.cols - 1 - N; j++)  {  //第一个通道  for (w = 0; w < N; w++)  {  for (h = 0; h < N; h++)  {  bb[w][h] = (int)src.at<Vec3b>(i - N / 2 + w, j - N / 2 + h)[0];  }  }  //comp = compute(bb);  //if (comp<0)comp = comp\*(-1);  //if (comp>255)comp = 255;  temp = ((int)src.at<Vec3b>(i, j)[0] + compute(bb)\*0.25);  temp = temp>255 ? 255 : temp;  temp = temp<0 ? 0 : temp;  src.at<Vec3b>(i, j)[0] = (int)temp;  //第二个通道  for (w = 0; w < N; w++)  {  for (h = 0; h < N; h++)  {  bb[w][h] = (int)src.at<Vec3b>(i - N / 2 + w, j - N / 2 + h)[1];  }  }  //comp = compute(bb);  //if (comp<0)comp = comp\*(-1);  //if (comp>255)comp = 255;  temp = ((int)src.at<Vec3b>(i, j)[1] + compute(bb)\*0.25);  temp = temp>255 ? 255 : temp;  temp = temp<0 ? 0 : temp;  src.at<Vec3b>(i, j)[1] = (int)temp;  //第三个通道  for (w = 0; w < N; w++)  {  for (h = 0; h < N; h++)  {  bb[w][h] = (int)src.at<Vec3b>(i - N / 2 + w, j - N / 2 + h)[2];  }  }  //comp = compute(bb);  //if (comp<0)comp = comp\*(-1);  //if (comp>255)comp = 255;  temp = ((int)src.at<Vec3b>(i, j)[2] + compute(bb)\*0.25);  temp = temp>255 ? 255 : temp;  temp = temp<0 ? 0 : temp;  src.at<Vec3b>(i, j)[2] = (int)temp;  }  }  return src;  }  void Laplacian()  {  arr[0][0] = 0;  arr[0][1] = 1;  arr[0][2] = 0;  arr[1][0] = 1;  arr[1][1] = -4;  arr[1][2] = 1;  arr[2][0] = 0;  arr[2][1] = 1;  arr[2][2] = 0;  }  void Robert()  {  //x方向  arr[0][0] = 0;  arr[0][1] = 0;  arr[0][2] = 0;  arr[1][0] = 0;  arr[1][1] = 1;  arr[1][2] = 0;  arr[2][0] = 0;  arr[2][1] = 0;  arr[2][2] = -1;  }  void Sobel()  {  //y方向  arr[0][0] = -1;  arr[0][1] = -2;  arr[0][2] = -1;  arr[1][0] = 0;  arr[1][1] = 0;  arr[1][2] = 0;  arr[2][0] = 1;  arr[2][1] = 2;  arr[2][2] = 1;  }  double compute(int bb[][N])  {  double temp = 0;  for (int i = 0; i < N; i++)  {  for (int j = 0; j < N; j++)  {  temp = arr[i][j] \* bb[i][j] + temp;  }  }  max1 = max1>temp ? max1 : temp;  min1 = max1<temp ? min1 : temp;  return temp;  }  void print()  {  for (int i = 0; i < N; i++)  {  for (int j = 0; j < N; j++)  {  cout << setw(3) << arr[i][j];  }  cout << endl;  }  cout << endl;  } |