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14日

|  |  |
| --- | --- |
| **实验名称** | 图像恢复 |
| **实验内容** | 1、均值滤波 具体内容：利用OpenCV对灰度图像像素进行操作，分别利用算术均值滤波器、几何均值滤波器、谐波和逆谐波均值滤波器进行图像去噪。模板大小为5\*5。（注：请分别为图像添加高斯噪声、胡椒噪声、盐噪声和椒盐噪声，并观察滤波效果）  2、中值滤波 具体内容：利用OpenCV对灰度图像像素进行操作，分别利用5\*5和9\*9尺寸的模板对图像进行中值滤波。（注：请分别为图像添加胡椒噪声、盐噪声和椒盐噪声，并观察滤波效果）  3、自适应均值滤波。 具体内容：利用OpenCV对灰度图像像素进行操作，设计自适应局部降低噪声滤波器去噪算法。模板大小7\*7（对比该算法的效果和均值滤波器的效果）  4、自适应中值滤波 具体内容：利用OpenCV对灰度图像像素进行操作，设计自适应中值滤波算法对椒盐图像进行去噪。模板大小7\*7（对比中值滤波器的效果）  5、彩色图像均值滤波 具体内容：利用OpenCV对彩色图像RGB三个通道的像素进行操作，利用算术均值滤波器和几何均值滤波器进行彩色图像去噪。模板大小为5\*5。 |
| **实验完成情况** | 核心代码见下面 |
| （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** | 在自适应均值滤波中，需要求图像的方差和模板的方差（meanStdDev），这个计算量非常大。所以速度很慢。  自适应中值滤波中，求出图像的最大值和最小值，再用if判断就可以了，但是要是它与最值相等要增大模板这个非常浪费时间，所以我在实现的时候直接返回了均值accumulate。  均值和中值都很简单，直接对模板排序就好了，用C++中的vector很容易。  试验中一5\*5为例子，同理7\*7改一下参数就可以了。 |
| （包括在实验 |
| 中遇到的问 |
| 题，以及解 |
| 决问题的方 |
| 法） |
| **实验结果** | 源文件见打包。 |
| （实验完成后 |
| 的源码和打 |
| 包文件的说 |
| 明） |

|  |
| --- |
| #include <opencv\highgui.h>  #include <cv.h>  #include <algorithm>  #include <numeric>  #include <vector>  #define N 5  #define path "..\\..\\pic\\zs\\y.jpg"  using namespace cv;  using namespace std;  void JZ55SS(Mat);  void JZ55JH(Mat);  void JZ55XB(Mat);  void JZ55NXB(Mat);  void JZ55ZZ(Mat);  void JZ55CS(Mat);  void ZSY55JZ(Mat);  void ZSY55ZZ(Mat);  int computeZSY55ZZ(vector <int>);  int computeZSY55JZ(vector <int>,Mat);  int computeSS(vector <int>);  int computeJH(vector <int>);  int computeXB(vector <int>);  int computeNXB(vector <int>);  int computeZZ(vector <int>);  int computeCS(vector <int>);  int main()  {  /\*根据电脑不同每次最好只做两个处理\*/  Mat src = imread(path);  imshow("原图", src);  //JZ55CS(src);//彩色均值  cvtColor(src, src, CV\_BGR2GRAY);//变换成灰度图像  //JZ55SS(src);//算数均值  //JZ55JH(src);//几何均值  //JZ55XB(src);//谐波  //JZ55NXB(src);//逆谐波  //JZ55ZZ(src);//中值滤波  //ZSY55JZ(src);//自适应均值  //ZSY55ZZ(src);//自适应中值  waitKey(0);  return 0;  }  void ZSY55ZZ(Mat src)  {  vector <int>vec;  int ii, jj;  int i = src.rows, j = src.cols;  int w = 0, h = 0;  cout << i << "\*" << j;  for (i = 0; i < src.rows; i++)  {  for (j = 0; j < src.cols; j++)  {  for (w = 0; w < N; w++)  {  for (h = 0; h < N; h++)  {  ii = i - N / 2 + w;  jj = j - N / 2 + h;  if (ii >= 0 && ii < src.rows&&jj >= 0 && jj < src.cols)  vec.push\_back((int)src.at<uchar>(ii, jj));  else  vec.push\_back(0);  //vec.push\_back((int)src.at<uchar>(i - 1 + w, j - 1 + h));  }  }  src.at<uchar>(i, j) = (uchar)computeZSY55ZZ(vec);  vec.clear();  }  }  imshow("自适应中值", src);  }  void ZSY55JZ(Mat src)  {  vector <int>vec;  int ii, jj;  int i = src.rows, j = src.cols;  int w = 0, h = 0;  cout << i << "\*" << j;  for (i = 0; i < src.rows; i++)  {  for (j = 0; j < src.cols; j++)  {  for (w = 0; w < N; w++)  {  for (h = 0; h < N; h++)  {  ii = i - N / 2 + w;  jj = j - N / 2 + h;  if (ii >= 0 && ii < src.rows&&jj >= 0 && jj < src.cols)  vec.push\_back((int)src.at<uchar>(ii, jj));  else  vec.push\_back(0);  //vec.push\_back((int)src.at<uchar>(i - 1 + w, j - 1 + h));  }  }  src.at<uchar>(i, j) = (uchar)computeZSY55JZ(vec,src);  vec.clear();  }  }  imshow("自适用均值", src);  }  void JZ55CS(Mat src)  {  vector <int>vec;  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++)  {  for (w = 0; w < N; w++)  {  for (h = 0; h < N; h++)  {  vec.push\_back((int)src.at<Vec3b>(i - 1 + w, j - 1 + h)[0]);    }  }  src.at<Vec3b>(i, j )[0] = (uchar)computeCS(vec);  vec.clear();  for (w = 0; w < N; w++)  {  for (h = 0; h < N; h++)  {  vec.push\_back((int)src.at<Vec3b>(i - 1 + w, j - 1 + h)[1]);  }  }  src.at<Vec3b>(i, j)[1] = (uchar)computeCS(vec);  vec.clear();  for (w = 0; w < N; w++)  {  for (h = 0; h < N; h++)  {  vec.push\_back((int)src.at<Vec3b>(i - 1 + w, j - 1 + h)[2]);  }  }  src.at<Vec3b>(i, j)[2] = (uchar)computeCS(vec);  vec.clear();  }  }  imshow("中值滤波", src);  }  void JZ55ZZ(Mat src)  {  vector <int>vec;  int ii, jj;  int i = src.rows, j = src.cols;  int w = 0, h = 0;  cout << i << "\*" << j;  for (i = 0; i < src.rows; i++)  {  for (j = 0; j < src.cols; j++)  {  for (w = 0; w < N; w++)  {  for (h = 0; h < N; h++)  {  ii = i - N / 2 + w;  jj = j - N / 2 + h;  if (ii >= 0 && ii < src.rows&&jj >= 0 && jj < src.cols)  vec.push\_back((int)src.at<uchar>(ii, jj));  else  vec.push\_back(0);  //vec.push\_back((int)src.at<uchar>(i - 1 + w, j - 1 + h));  }  }  src.at<uchar>(i, j) = (uchar)computeZZ(vec);  vec.clear();  }  }  imshow("中值滤波", src);  }  void JZ55NXB(Mat src)  {  vector <int>vec;  int ii, jj;  int i = src.rows, j = src.cols;  int w = 0, h = 0;  cout << i << "\*" << j;  for (i = 0; i < src.rows; i++)  {  for (j = 0; j < src.cols; j++)  {  for (w = 0; w < N; w++)  {  for (h = 0; h < N; h++)  {  ii = i - N / 2 + w;  jj = j - N / 2 + h;  if (ii >= 0 && ii < src.rows&&jj >= 0 && jj < src.cols)  vec.push\_back((int)src.at<uchar>(ii, jj));  else  vec.push\_back(0);  //vec.push\_back((int)src.at<uchar>(i - 1 + w, j - 1 + h));  }  }  src.at<uchar>(i, j) = (uchar)(computeNXB(vec) \* 7);  vec.clear();  }  }  imshow("逆谐波", src);  }  void JZ55XB(Mat src)  {  vector <int>vec;  int ii, jj;  int i = src.rows, j = src.cols;  int w = 0, h = 0;  cout << i << "\*" << j;  for (i = 0; i < src.rows; i++)  {  for (j = 0; j < src.cols; j++)  {  for (w = 0; w < N; w++)  {  for (h = 0; h < N; h++)  {  ii = i - N / 2 + w;  jj = j - N / 2 + h;  if (ii >= 0 && ii < src.rows&&jj >= 0 && jj < src.cols)  vec.push\_back((int)src.at<uchar>(ii, jj));  else  vec.push\_back(0);  //vec.push\_back((int)src.at<uchar>(i - 1 + w, j - 1 + h));  }  }  src.at<uchar>(i, j) = (uchar)computeXB(vec)\*5;  vec.clear();  }  }  imshow("谐波", src);  }  void JZ55JH(Mat src)  {  vector <int>vec;  int ii, jj;  int i = src.rows, j = src.cols;  int w = 0, h = 0;  cout << i << "\*" << j;  for (i = 0; i < src.rows; i++)  {  for (j = 0; j < src.cols; j++)  {  for (w = 0; w < N; w++)  {  for (h = 0; h < N; h++)  {  ii = i - N / 2 + w;  jj = j - N / 2 + h;  if (ii >= 0 && ii < src.rows&&jj >= 0 && jj < src.cols)  vec.push\_back((int)src.at<uchar>(ii, jj));  else  vec.push\_back(0);  //vec.push\_back((int)src.at<uchar>(i - 1 + w, j - 1 + h));  }  }  src.at<uchar>(i, j) = (uchar)computeJH(vec);  vec.clear();  }  }  imshow("几何均值", src);  }  void JZ55SS(Mat src)  {  vector <int>vec;  int ii, jj;  int i = src.rows, j = src.cols;  int w = 0, h = 0;  cout << i << "\*" << j;  for (i = 0; i < src.rows; i++)  {  for (j = 0; j < src.cols; j++)  {  for (w = 0; w < N; w++)  {  for (h = 0; h < N; h++)  {  ii = i - N / 2 + w;  jj = j - N / 2 + h;  if (ii >= 0 && ii < src.rows&&jj >= 0 && jj < src.cols)  vec.push\_back((int)src.at<uchar>(ii, jj));  else  vec.push\_back(0);  //vec.push\_back((int)src.at<uchar>(i - 1 + w, j - 1 + h));  }  }  src.at<uchar>(i,j)=(uchar)computeSS(vec);  vec.clear();  }  }  imshow("算数均值", src);  }  int computeSS(vector <int>vec)  {  int sum = 0;  for (int i = 0; i < vec.size(); i++)  {  sum += vec[i];  }  return sum / (N\*N);  }  int computeJH(vector <int>vec)  {  double sum = 1.0;  //cout << vec.size();  for (int i = 0; i < vec.size(); i++)  {  if (vec[i] != 0)  sum = vec[i] \* sum;  }  return (int)pow(sum,1.0/(N\*N));  }  int computeXB(vector <int>vec)  {  double sum = 1.0;  //cout << vec.size();    for (int i = 0; i < vec.size(); i++)  {  if (vec[i] != 0)  sum = 1.0/vec[i] + sum;  }  return (1.0\*N\*N)/sum;  }  int computeNXB(vector <int>vec)  {  double sum = 1.0;  //cout << vec.size();  for (int i = 0; i < vec.size(); i++)  {  if (vec[i] != 0)  sum = 1.0 / vec[i] + sum;  }  return (1.0\*N\*N) / sum;  }  int computeZZ(vector <int>vec)  {  sort(vec.begin(),vec.end());  return vec[(N\*N)/2+1];  }  int computeCS(vector <int>vec)  {  int sum = 0;  for (int i = 0; i < vec.size(); i++)  {  sum += vec[i];  }  return sum / (N\*N);  }  int computeZSY55JZ(vector <int>vec,Mat src)  {  //Mat src = imread(path);  double jz, fc;  double k = 1;  Scalar mean;  Scalar stddev;  meanStdDev(src, mean, stddev);  //jz = mean.val[0];  fc = stddev.val[0];  double sum = accumulate(begin(vec), end(vec), 0.0);  double mean1 = sum / vec.size(); //均值  double accum = 0.0;  for\_each(begin(vec),end(vec), [&](const double d)  {  accum += (d - mean1)\*(d - mean1);  });  double stdev1 = sqrt(accum / (vec.size() - 1)); //方差  k = (stdev1 > fc) ? (fc / stdev1) : (stdev1 / fc);  return vec[(N\*N) / 2 + 1]-k\*(vec[(N\*N) / 2 + 1] - mean1);  }  int computeZSY55ZZ(vector <int>vec)  {  int med;  int min = 0;  int max = 255;  sort(vec.begin(), vec.end());  med = vec[(N\*N) / 2 + 1];  if (med > vec[0]&&med < vec[vec.size()-1])  {  return med;  }  else  {  if (med>0 && med < 255)  {  return med;  }  else  {  double sum = accumulate(begin(vec), end(vec), 0.0);  return sum/(N\*N);  }  }  } |