|  |  |
| --- | --- |
| **实验名称** | **实验 5：频域滤波** |
| **实验目的** | 掌握图像进行频域滤波的方法和步骤。  1、掌握图像频域 DFT 变换和反变换的方法。  2、掌握图像频域滤波的步骤 |
| **实验内容** | 1、灰度图像的 DFT 和 IDFT。  具体内容：利用 OpenCV 提供的 cvDFT 函数对图像进行 DFT 和 IDFT 变换  2、利用理想高通和低通滤波器对灰度图像进行频域滤波  具体内容：利用 cvDFT 函数实现 DFT，在频域上利用理想高通和低通滤波器进行滤波，并把滤波过后的图像显示在屏幕上（观察振铃现象），要求截止频率可输入。  3、利用布特沃斯高通和低通滤波器对灰度图像进行频域滤波。  具体内容：利用 cvDFT 函数实现 DFT，在频域上进行利用布特沃斯高通和低通滤波器进行滤波，并把滤波过后的图像显示在屏幕上（观察振铃现象），要求截止频率和 n 可输入。 |
| **实验完成情况** | 1. 实验步骤：利用 OpenCV 提供的 cvDFT 函数对图像进行 DFT 和 IDFT 变换   核心代码如下：  //DFT变换  IplImage \*DFT(IplImage \* src)  {  IplImage\* fourier = cvCreateImage(cvGetSize(src),IPL\_DEPTH\_64F,2);  int dft\_H, dft\_W;  dft\_H = src->height;  dft\_W = src->width;  CvMat \*src\_Re = cvCreateMat(dft\_H,dft\_W, CV\_64FC1); // double Re, Im;  CvMat \*src\_Im = cvCreateMat(dft\_H,dft\_W, CV\_64FC1); //Imaginary part  CvMat \*sum\_src =cvCreateMat(dft\_H,dft\_W, CV\_64FC2); //2 channels (src\_Re, src\_Im)  CvMat \*sum\_dst =cvCreateMat(dft\_H,dft\_W, CV\_64FC2); //2 channels (dst\_Re, dst\_Im)  cvConvert(src, src\_Re);  cvZero(src\_Im);  cvMerge(src\_Re, src\_Im, 0, 0, sum\_src);  cvDFT(sum\_src,sum\_dst,CV\_DXT\_FORWARD,0);  cvConvert(sum\_dst, fourier);  cvReleaseMat(&src\_Re);  cvReleaseMat(&src\_Im);  cvReleaseMat(&sum\_src);  cvReleaseMat(&sum\_dst);  return fourier;  }  //DFT反变换  IplImage \*IDFT(IplImage \* fourier)  {  IplImage\* dst = cvCreateImage(cvGetSize(fourier),IPL\_DEPTH\_8U,1);  int dft\_H, dft\_W;  dft\_H = fourier->height;  dft\_W = fourier->width;  CvMat \*dst\_Re = cvCreateMat(dft\_H,dft\_W, CV\_64FC1); // double Re, Im;  CvMat \*dst\_Im = cvCreateMat(dft\_H,dft\_W, CV\_64FC1); //Imaginary part  CvMat \*sum\_dst =cvCreateMat(dft\_H,dft\_W, CV\_64FC2); //2 channels (dst\_Re, dst\_Im)  CvMat \*sum\_src = cvCreateMat(dft\_H,dft\_W, CV\_64FC2 );  cvConvert(fourier, sum\_src);  cvDFT(sum\_src,sum\_dst,CV\_DXT\_INV\_SCALE,0);  cvSplit(sum\_dst,dst\_Re,dst\_Im,0,0);  cvConvert(dst\_Re, dst);  cvReleaseMat(&dst\_Re);  cvReleaseMat(&dst\_Im);  cvReleaseMat(&sum\_src);  cvReleaseMat(&sum\_dst);  return dst;  }  //归一化，将灰度映射到0~255之间, 并将能量最高的四角移到中心, 生成图片频域能量图  void BuildDFTImage(IplImage \*fourier, IplImage \*dst)  {  IplImage \*image\_Re = 0, \*image\_Im = 0;  image\_Re = cvCreateImage(cvGetSize(fourier), IPL\_DEPTH\_64F, 1);  image\_Im = cvCreateImage(cvGetSize(fourier), IPL\_DEPTH\_64F, 1); //Imaginary part    cvSplit(fourier, image\_Re, image\_Im, 0, 0 );    // Compute the magnitude of the spectrum Mag = sqrt(Re^2 + Im^2)  cvPow( image\_Re, image\_Re, 2.0);  cvPow( image\_Im, image\_Im, 2.0);  cvAdd( image\_Re, image\_Im, image\_Re);  cvPow( image\_Re, image\_Re, 0.5 );    cvReleaseImage(&image\_Im);  cvAddS(image\_Re, cvScalar(1.0), image\_Re); // 1 + Mag  cvLog(image\_Re, image\_Re ); // log(1 + Mag)    //重新安排傅里叶图像中心  // Rearrange the quadrants of Fourier image so that the origin is at  // the image center  double minVal = 0, maxVal = 0;  cvMinMaxLoc( image\_Re, &minVal, &maxVal ); // Localize minimum and maximum values    CvScalar min;  min.val[0] = minVal;  double scale = 255 / (maxVal - minVal);  cvSubS(image\_Re, min, image\_Re);  cvConvertScale(image\_Re, dst, scale);  cvReleaseImage(&image\_Re);  // Rearrange the quadrants of Fourier image so that the origin is at  // the image center  int nRow, nCol, i, j, cy, cx;  uchar tmp13, tmp24;  nRow = fourier->height;  nCol = fourier->width;  cy = nRow/2; // image center  cx = nCol/2;  for( j = 0; j < cy; j++ )  {  for( i = 0; i < cx; i++ )  {  tmp13 = CV\_IMAGE\_ELEM( dst, uchar, j, i);  CV\_IMAGE\_ELEM( dst, uchar, j, i) = CV\_IMAGE\_ELEM(dst, uchar, j+cy, i+cx);  CV\_IMAGE\_ELEM( dst, uchar, j+cy, i+cx) = tmp13;    tmp24 = CV\_IMAGE\_ELEM( dst, uchar, j, i+cx);  CV\_IMAGE\_ELEM( dst, uchar, j, i+cx) = CV\_IMAGE\_ELEM( dst, uchar, j+cy, i);  CV\_IMAGE\_ELEM( dst, uchar, j+cy, i) = tmp24;  }  }  }  实验结果如图：       1. 实验步骤：利用 cvDFT 函数实现 DFT，在频域上利用理想高通和低通滤波器进行滤波，并把滤波过后的图像显示在屏幕上（观察振铃现象），截止频率可输入。   核心代码如下：  void PassFilter(IplImage \* fourier, int FLAG, double d0, int n1)  {  int i, j;  int state = -1;  double tempD;  long width, height;  width = fourier->width;  height = fourier->height;  long x, y;  x = width / 2;  y = height / 2;  CvMat\* H\_mat;  H\_mat = cvCreateMat(fourier->height,fourier->width, CV\_64FC2);  for(i = 0; i < height; i++){  for(j = 0; j < width; j++){  if(i > y && j > x){  state = 3;  }else if(i > y){  state = 1;  }else if(j > x){  state = 2;  }else{  state = 0;  }    switch(state){  case 0:  tempD = (double)sqrt(1.0\*i \* i + j \* j);break;  case 1:  tempD = (double)sqrt(1.0\*(height - i) \* (height - i) + j \* j);break;  case 2:  tempD = (double)sqrt(1.0\*i \* i + (width - j) \* (width - j));break;  case 3:  tempD = (double)sqrt(1.0\*(height - i) \* (height - i) + (width - j) \* (width - j));break;  default:  break;  }  switch(FLAG){  case IDEAL\_LOW:  if(tempD <= D0){  ((double\*)(H\_mat->data.ptr + H\_mat->step \* i))[j \* 2] = 1.0;  ((double\*)(H\_mat->data.ptr + H\_mat->step \* i))[j \* 2 + 1] = 0.0;  }else{  ((double\*)(H\_mat->data.ptr + H\_mat->step \* i))[j \* 2] = 0.0;  ((double\*)(H\_mat->data.ptr + H\_mat->step \* i))[j \* 2 + 1] = 0.0;  }  break;  case IDEAL\_HIGH:  if(tempD <= D0){  ((double\*)(H\_mat->data.ptr + H\_mat->step \* i))[j \* 2] = 0.0;  ((double\*)(H\_mat->data.ptr + H\_mat->step \* i))[j \* 2 + 1] = 0.0;  }else{  ((double\*)(H\_mat->data.ptr + H\_mat->step \* i))[j \* 2] = 1.0;  ((double\*)(H\_mat->data.ptr + H\_mat->step \* i))[j \* 2 + 1] = 0.0;  }  break;  case BW\_LOW:  tempD = 1 / (1 + pow(tempD / D0, 2 \* n));  ((double\*)(H\_mat->data.ptr + H\_mat->step \* i))[j \* 2] = tempD;  ((double\*)(H\_mat->data.ptr + H\_mat->step \* i))[j \* 2 + 1] = 0.0;  break;  case BW\_HIGH:  tempD = 1 / (1 + pow(D0 / tempD, 2 \* n));  ((double\*)(H\_mat->data.ptr + H\_mat->step \* i))[j \* 2] = tempD;  ((double\*)(H\_mat->data.ptr + H\_mat->step \* i))[j \* 2 + 1] = 0.0;  break;  default:  break;  }  }  }  cvMulSpectrums(fourier, H\_mat, fourier, CV\_DXT\_ROWS);  cvReleaseMat(&H\_mat);  }  其中选择IDEAL\_LOW与IDEAL\_HIGH模式即为该实验的理想低通与高通滤波器  实验结果如图：    理想低通滤波器处理过程如下图所示      理想高通滤波器处理过程如下图所示      3、实验步骤：利用 cvDFT 函数实现 DFT，在频域上进行利用布特沃斯高通和低通滤波器进行滤波，并把滤波过后的图像显示在屏幕上（观察振铃现象），并且截止频率和 n 可输入。  核心代码与2中类似  其中选择BW\_LOW与BW\_HIGH模式即为该实验的理想低通与高通滤波器  实验结果如图：  巴特沃斯低通滤波器处理过程如下图所示      巴特沃斯高通滤波器处理过程如下图所示 |
| **实验中的问题** | 问题：DFT变换后图像全为黑色。  解决方法：DFT处理后未调整图像中心，高能量集中在图片的四个拐角，重新设计算法中心化该谱，将能量集中到图像中心。 |
| **实验结果** | 实验源码位于 实验5\_曹欣\_SA14225013.rar中  代码位于lab5文件夹中，5\_1.cpp为主程序，可在宏定义中修改截止频率D0 |