**版本信息**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版本信息记录表** | | | | |
| 版本号 | 修订内容 | 修订人 | 修订日期 | 审核人 |
| 1.0 | 初版 | 党慧杰 | 2018.08.20 |  |
| 2.0 | 修订 | 党慧杰 | 2018.10.15 |  |
|  |  |  |  |  |

# 图像滤波

## 【实验目的】

1、从读取Kinect图像，对图像进行滤波并显示结果

2、学习使用了均值滤波和中值滤波

## 【实验原理】

**1. OpenCV简介**

OpenCV是一个开放源代码的计算机视觉应用平台，由英特尔公司下属研发中心俄罗斯团队发起该项目，开源BSD证书，OpenCV的目标是实现实时计算机视觉，是一个跨平台的计算机视觉库。其应用领域包括：二维和三维特征工具箱、运动估算、人脸系统识别、姿势识别、人机交互、移动机器人、运动理解、对象鉴别、分割与识别、立体视觉、运动跟踪、增强实现(AR技术)。

图像滤波，即在尽量保留图像细节特征的条件下对目标图像的噪声进行抑制，是图像预处理中不可缺少的操作，其处理效果的好坏将直接影响到后续图像处理和分析的有效性和可靠性。

**2. 均值滤波**

均值滤波是典型的线性滤波算法，主要方法为邻域平均法，即用一片图像区域的各个像素的均值来代替原图像中的各个像素值。一般需要在图像上对目标像素给出一个模板（内核），该模板包括了其周围的临近像素（比如以目标像素为中心的周围8（3x3-1）个像素，构成一个滤波模板，即去掉目标像素本身）。再用模板中的全体像素的平均值来代替原来像素值。即对待处理的当前像素点（x，y），选择一个模板，该模板由其近邻的若干像素组成，求模板中所有像素的均值，再把该均值赋予当前像素点（x，y），作为处理后图像在该点上的灰度个g（x，y），即个g（x，y）=1/m ∑f（x，y） ，其中m为该模板中包含当前像素在内的像素总个数。

函数原型：

void blur(InputArray src, OutputArray dst, Size ksize, Point anchor=Point(-1,-1), int borderType=BORDER\_DEFAULT )

参数说明：

* src: 输入图像
* dts: 结果输出图像
* ksize: 内核大小
* anchor: 锚点，即被平滑的点
* boderType: 推断图像外部像素的边界模式

**3. 中值滤波**

中值滤波器是一种非线性滤波器，常用于消除图像中的椒盐噪声。与低通滤波不同的是，中值滤波有利于保留边缘的尖锐度，但它会洗去均匀介质区域中的纹理。

在输入图像x(n1, n2)中，以任一像素为中心设置一个确定的邻域A，A的边长为2N+1,（N=0,1,2,…）。将邻域内各像素的强度值按大小顺序排列，取位于中间位置的那个值（中值）作为该像素点的输出值，遍历整幅图像就可完成整个滤波过程：A=x(i,j),  y=Med{x1, x2, x3,…,x2N+1}

函数原型：

void medianBlur(InputArray src, OutputArray dst, int ksize)

参数说明：

* src: 输入图像
* dst: 输出图像, 必须与src相同类型
* ksize: 内核大小 (只需一个值，因为使用正方形窗口)，必须为奇数

## 【实验环境】

操作系统

Windows 系列 36位/64位

开发环境

Visual Studio 2012、OpenCV

开发语言

C++

## 【实验步骤】

### 一、新建Win32控制台应用程序

1. 点击菜单栏【文件】->【新建】->【项目】或快捷件Ctrl + Shift + N 新建项目，如下图所示：



图 1

2. 在弹出的【新建项目】框中选择模板Visual C++，在中间的栏目中选择【Win32 控制台应用程序】。输入工程名称和保存路径，点击确定。

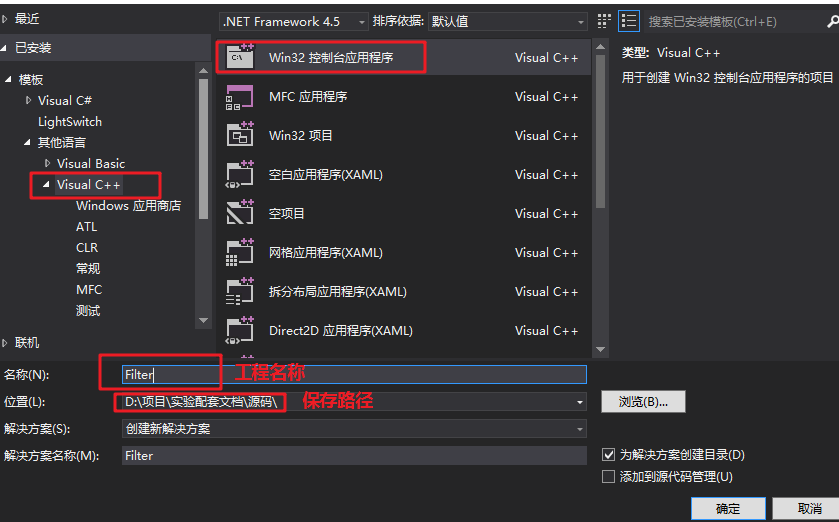


图 2

3. 在弹出的【Win32 应用程序向导】对话框中，保持默认配置，直至完成。

### 二、OpenCV配置

在Visual Studio 2012项目中配置OpenCV，参见文档《Opencv的配置》。Kinect的配置，参见文档《Kinect使用说明》。

### 三、均值滤波的实现

**1. 新建ImageHandler类**

1）右击项目，选择【添加】->【类】，如下图所示：

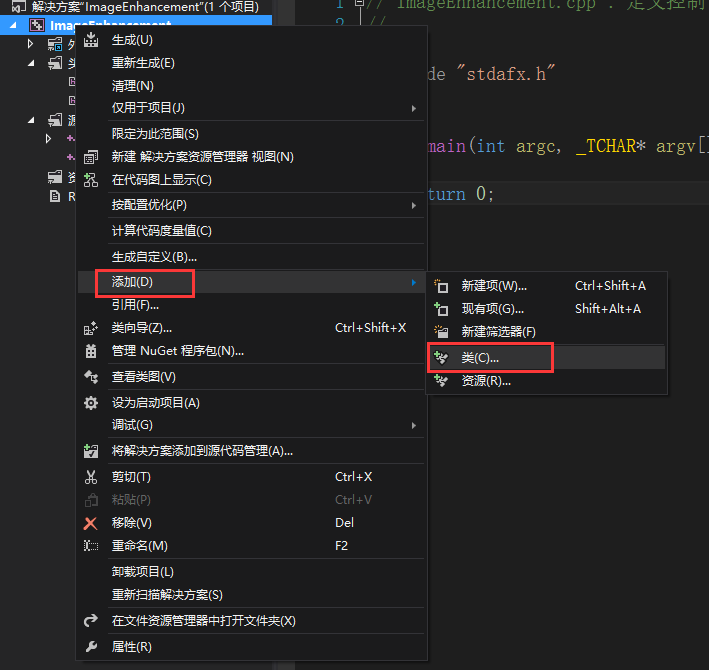


图3

2）选择C++类，点击【添加】

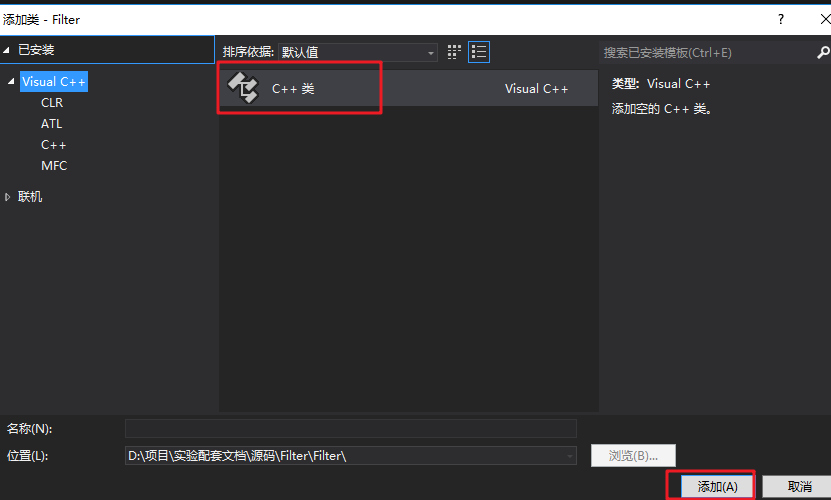


图 4

3）输入类名，点击确定，新类就创建完成。

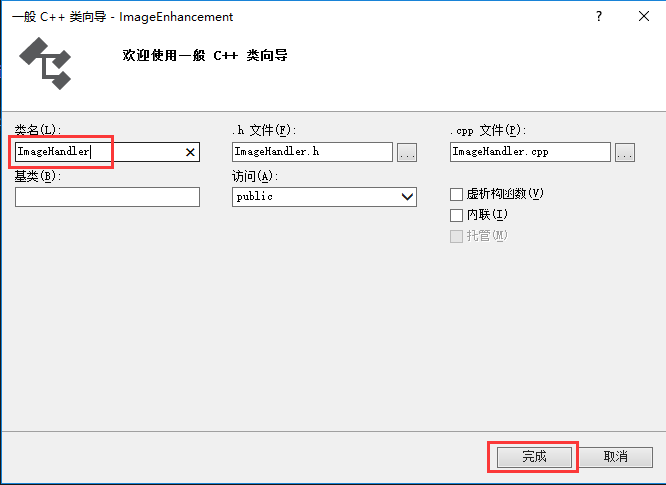


图 5

**2. 引入必需的头文件和命名空间**

该方法主要是利用OpenCV提供的函数来完成，因而首先必需添加相关的头文件。打开‘ImageHandler.h’文件。

添加如下头文件：

#include <opencv2/opencv.hpp>

引入命名空间：

using namespace cv;

如下图所示：

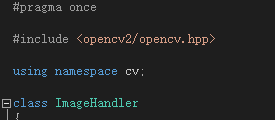


图6

注意：如果不能导入头文件，请检查OpenCV的配置是否正确。

**3. 对图像进行均值滤波**

1）声明函数

打开‘ImageHandler.h’文件，在 ImageHandler 类中添加如下函数声明：

void FilterMean(Mat frame);

其中，参数frame为需要处理的一帧图像。

2） 函数实现

算法实现的流程如下：

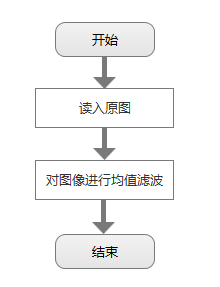


图7

打开‘ImageHandler.cpp’文件，键入如下程序：

void ImageHandler::FilterMean(Mat frame)

{

Mat resultImage;

blur(frame, resultImage, Size(5,5));

imshow("均值滤波", resultImage);

}

### 四、中值滤波的实现

**1. 对图像进行中值滤波**

1）声明函数

打开‘ImageHandler.h’文件，在 ImageHandler 类中添加如下函数声明：

void FilterMedium(Mat frame);

其中，参数frame为需要处理的一帧图像。

2）函数实现

算法实现流程如下：

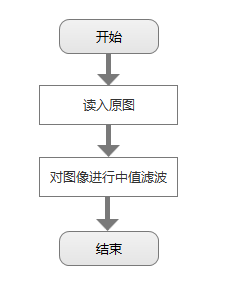


图 8

打开‘ImageHandler.cpp’文件，键入如下程序：

void ImageHandler::FilterMedium(Mat frame)

{

Mat resultImage;

medianBlur(frame, resultImage, 5);

imshow("中值滤波", resultImage);

}

### 五、从Kinect获取彩色图像并进行图像滤波处理

**1. 引入头文件**

在主程序入口函数之前，添加如下的头部文件：

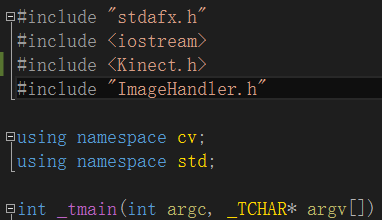


图 9

**2. 获取Kinect彩色图像**

从Kinect获取图像的一般过程为：

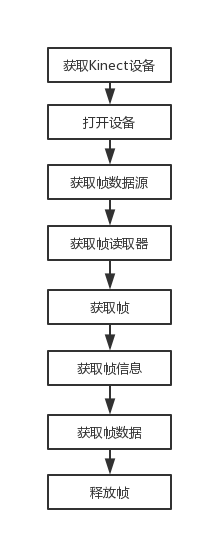


图 10

相关代码如下：

IKinectSensor\* pKinectSensor = NULL;

IColorFrameSource\* pColorFrameSource = NULL;

IColorFrameReader\* pColorFrameReader = NULL;

IFrameDescription\* pFrameDescription = NULL;

ColorImageFormat imgFormat = ColorImageFormat\_None;

int width, height;

uchar\* pBuffer = NULL;

UINT bufferSize = 0;

// 获取设备

HRESULT hr = GetDefaultKinectSensor(&pKinectSensor);

if (FAILED(hr))

{

cout << "获取Kinect设备失败" << endl;

goto exit;

}

// 打开设备

hr = pKinectSensor->Open();

if (FAILED(hr))

{

cout << "打开Kinect设备失败" << endl;

goto exit;

}

// 获取彩色帧数据源

hr = pKinectSensor->get\_ColorFrameSource(&pColorFrameSource);

if (FAILED(hr))

{

cout << "获取Kinect彩色数据源失败" << endl;

goto exit;

}

// 打开颜色帧读取器

hr = pColorFrameSource->OpenReader(&pColorFrameReader);

if (FAILED(hr))

{

cout << "打开Kinect颜色帧读取器失败" << endl;

goto exit;

}

while (1)

{

IColorFrame\* pColorFrame = NULL;

hr = pColorFrameReader->AcquireLatestFrame(&pColorFrame);

if (FAILED(hr)) continue;

// 获取帧信息

pColorFrame->get\_FrameDescription(&pFrameDescription);

pFrameDescription->get\_Width(&width);

pFrameDescription->get\_Height(&height);

// 获取图像格式

pColorFrame->get\_RawColorImageFormat(&imgFormat);

// 拷贝数据

Mat img(height, width, CV\_8UC4);

pBuffer = img.data;

bufferSize = img.rows \* img.step;

pColorFrame->CopyConvertedFrameDataToArray(bufferSize, reinterpret\_cast<BYTE\*>(pBuffer), ColorImageFormat\_Bgra);

// 释放帧

pColorFrame->Release();

imshow("Kinect彩色数据", img);

if (waitKey(30) == 27) // 按ESC退出

{

break;

}

}

**3. 对图像进行滤波处理**

1）在入口程序\_tmain中添加ImageHnadler实例对象。如下所示：

ImageHandler imgHandler;

2）在while（1）中，定位到‘释放帧’代码。在该代码下添加如下处理函数调用：

// 缩小尺寸

resize(img, img, cv::Size(width / 4, height / 4));

// 颜色转换

cvtColor(img, img, CV\_BGRA2BGR);

// 图像滤波

imgHandler.FilterMean(img);

imgHandler.FilterMedium(img);

**4. 运行程序**

点击vs工具栏中的【本地Windows调试器】（如下图）运行该程序。



图 11

程序成功运行后，将弹出如下窗口：

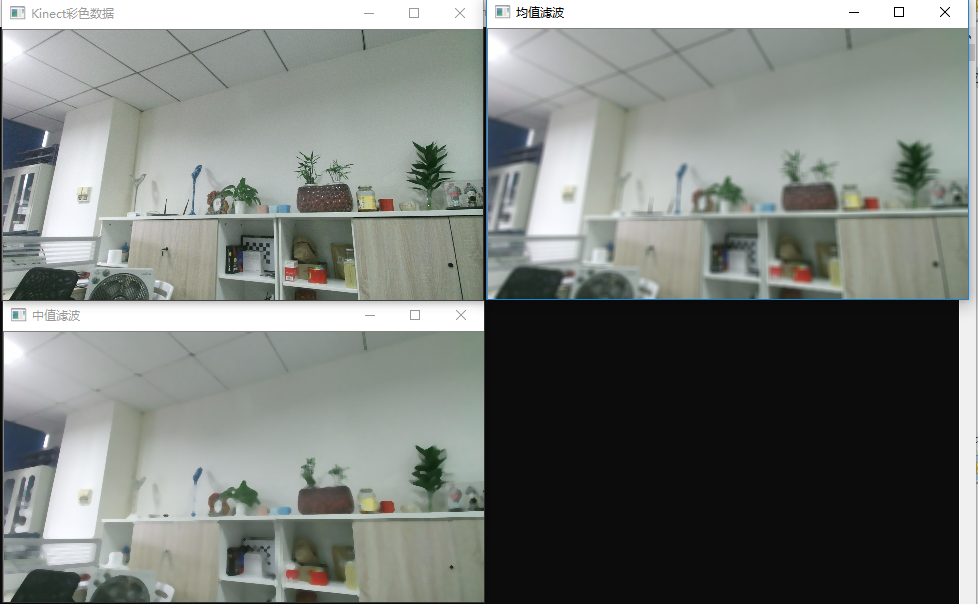


图 12

## 【思考题】

### 选择题

题目1：图像滤波算法包括：（D）

A、中值滤波

B、均值滤波

C、高斯滤波

D、以上都是

题目2：高通滤波后的图像通常较暗，为改善这种情况，将高通滤波器的转移函数加上一常数量以便引入一些低频分量。这样的滤波器叫？（ B ）

A、巴特沃斯高通滤波器

C、高频加强滤波器

B、高频提升滤波器

D、理想高通滤波器

### 2、简答题

题目1：中值滤波的实质是什么？

对受到噪声污染的图像可以采用线性滤波的方法来处理，但是很多线性滤波有低通性，在去噪声的同时也使得边缘模糊了，中值滤波在某些情况下可以做到既去除噪声又保护图像的边缘，它是一种非线性的去除噪声的方法。

中值滤波的实现原理是把数字图像中一点的值用该点的一个区域的各个点的值的中值代替。我们将一个点的特定长度或形状的邻域称为窗口，那么对于二维图像的中值滤波，一般采用3\*3或5\*5的窗口进行滤波。