**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 |  |
|  |  |  |  |  |

# 图像直方图与图像增强

## 【实验目的】

1、学习图像直方图均衡化、拉普拉斯图像增强和伽马变换图像增强。

2、读取图片，进行图像处理，并显示结果。

## 【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)。

**2. 基于直方图均衡化的图像增强**

直方图均衡化是通过调整图像的灰阶分布，使得在0~255灰阶上的分布更加均衡，提高了图像的对比度，达到改善图像主观视觉效果的目的。对比度较低的图像适合使用直方图均衡化方法来增强图像细节。是通过归一化直方图来规范灰度图像的亮度和对比度。

函数原型如下：

void equalizeHist(InputArray src, OutputArray dst)

参数说明：

* src: 需要处理的单通道图像；
* dst: 处理结果输出图像

**3. 基于拉普拉斯的图像增强算法**

通过选取合适的拉普拉斯算子与图像卷积达到锐化增强图像的目的。从而达到增强局部图像对比度。中心为5的8领域拉普拉斯算子如下所示：

|  |  |  |
| --- | --- | --- |
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

函数原型：



参数说明：

* src: 原始图像
* dst: 结果输出图像。
* ddepth: 目标图像深度。如果为负则保持与原始图像的深度一致。
* kernel: 卷积核，一个单通道浮点型矩阵。
* anchor: 内核的基准点，默认值为(-1,-1)。
* delta: 在存储目标图像前可选的添加到图像的值，默认为0。

borderType: 像素向外逼近的方法。默认为BORDER\_DEFAULT，即对全部边界进行计算。

**4. 基于伽马变换图像增强**

伽马变换主要用于图像的校正，将灰度过高或者灰度过低的图片进行修正，增强对比度。变换公式就是对原图像上每一个像素值做乘积运算：

伽马变换对图像的修正作用其实就是通过增强低灰度或高灰度的细节实现的。γ值以1为分界，值越小，对图像低灰度部分的扩展作用就越强；值越大，对图像高灰度部分的扩展作用就越强。通过不同的γ值，就可以达到增强低灰度或高灰度部分细节的作用。伽马变换对于图像对比度偏低，并且整体亮度值偏高（对于相机过曝）情况下的图像增强效果明显。

1. **流程图**

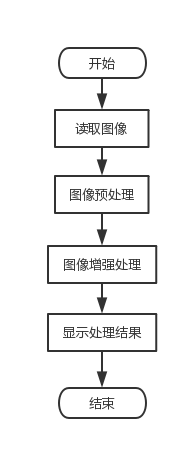
****

图1 处理流程

## 【实验环境】

操作系统

Windows 8及以上36位/64位

开发环境

Visual Studio 2012、OpenCV、Kinect

开发语言

C++

设备

PC（配备USB3.0）、Kinect

## 【实验步骤】

### 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 2

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。

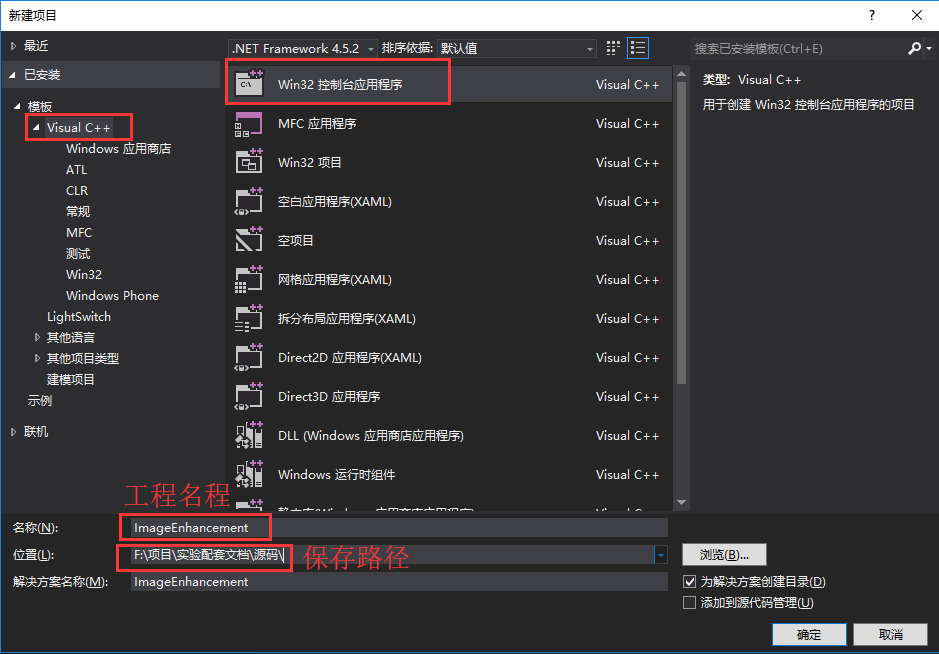


图3

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。

### 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》。

### 基于直方图均衡化的图像增强

**1. 新建ImageHandler类**

1）右击项目，选择【添加】->【类】，如下图所示：

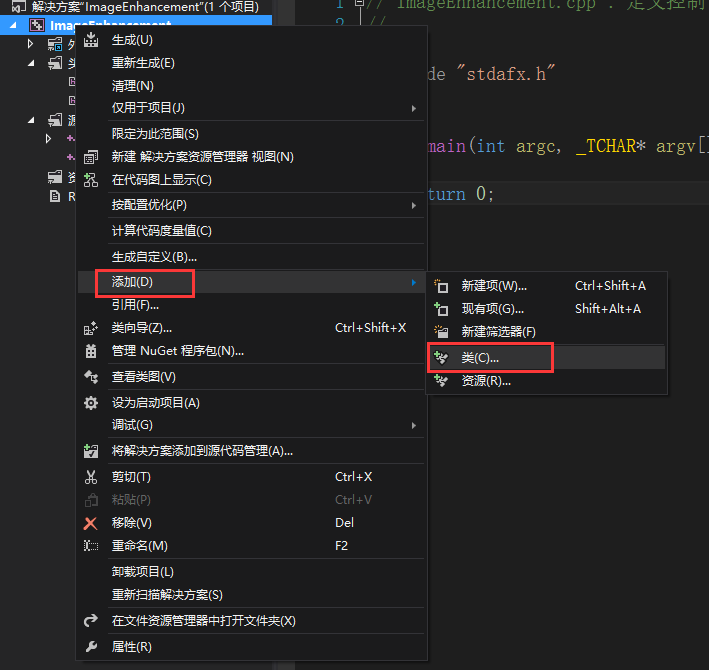


图 4

2）选择C++类，点击【添加】

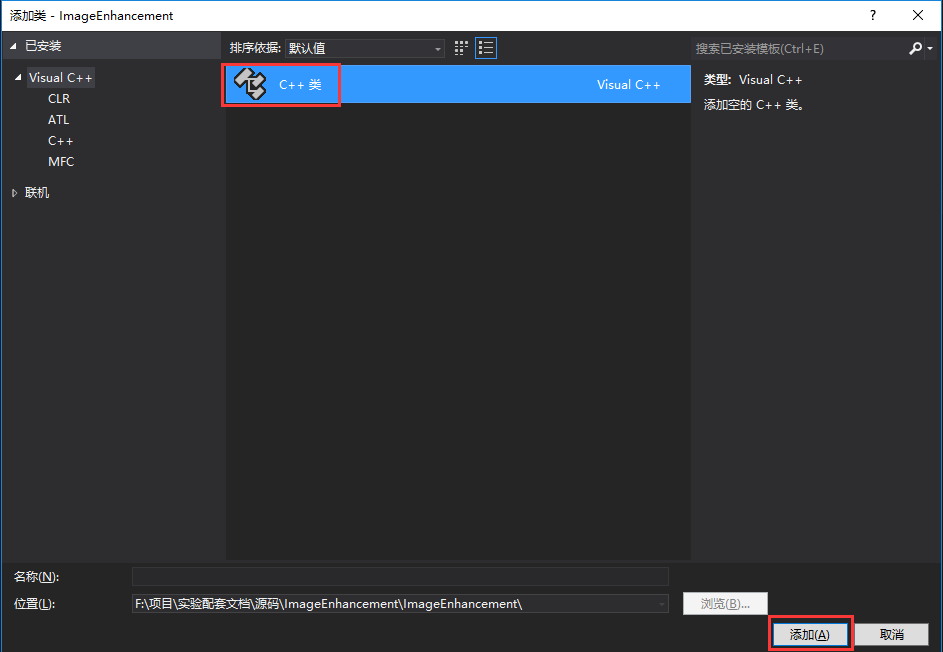


图 5

3）输入类名，点击确定，新类就创建完成。

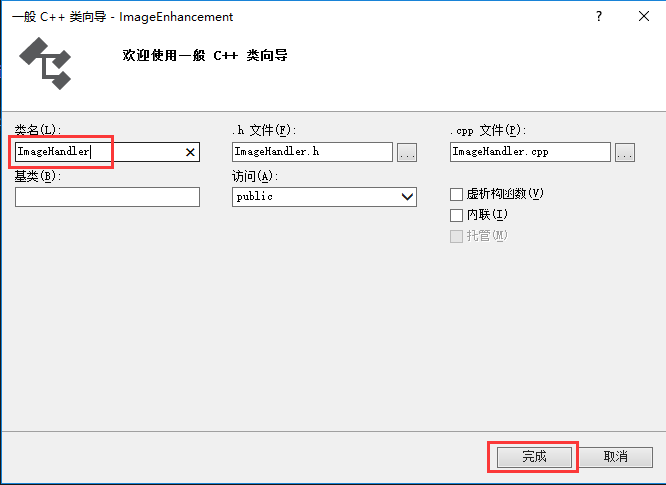


图6

**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 <iostream>

#include <opencv2/imgproc/imgproc.hpp>

#include <opencv2/highgui/highgui.hpp>

#include <opencv2/core/core.hpp>

引入命名空间：

using namespace cv;

如下图所示：

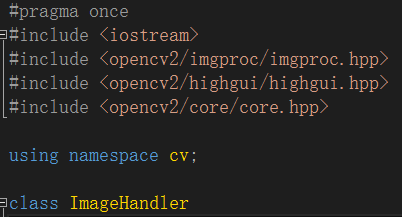


图7

注意：如果不能导入头文件，请检查OpenCV的配置是否正确。

**3. 图像直方图均衡化的实现**

1. 声明函数

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

void ImgEqualize(Mat frame);

其中，参数frame为需要处理的一帧图像。

1. 函数实现

算法实现的流程如下：

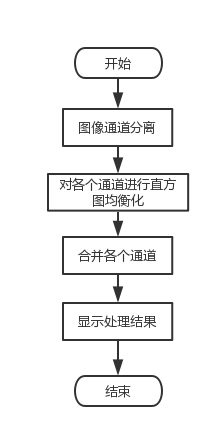


图8

打开‘ImageHandler.cpp’文件，键入如下程序：

void ImageHandler::ImgEqualize(Mat frame)

{

Mat resultImage;

Mat imageRGB[3];

// 图像通道分离

split(frame, imageRGB);

for (int i = 0; i < 3; i++)

{

// 直方图均衡化

equalizeHist(imageRGB[i], imageRGB[i]);

}

// 通道合并

merge(imageRGB, 3, resultImage);

// 显示处理结果

imshow("图像直方图均衡化", resultImage);

}

### 基于拉普拉斯的图像增强算法

在本实验中选择的卷积核为：

|  |  |  |
| --- | --- | --- |
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

1. **拉普拉斯的图像增强算法的实现**
2. 声明函数

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

void ImgEnhanceLaplace(Mat frame);

其中，参数frame为需要处理的一帧图像。

1. 函数实现

算法实现流程如下：

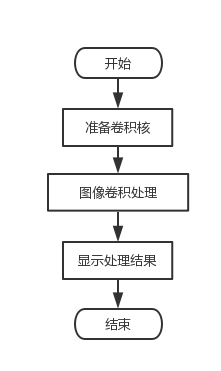


图 11

打开‘ImageHandler.cpp’文件，键入如下程序：

void ImageHandler::ImgEnhanceLaplace(Mat frame)

{

Mat resultImage;

// 卷积核, 单通道浮点矩阵

Mat kernel = (Mat\_<float>(3, 3) << 0, -1, 0, -1, 5, -1, 0, -1, 0);

// 图像卷积（图像滤波)

filter2D(frame, resultImage, CV\_8UC3, kernel);

imshow("拉普拉斯算子图像增强", resultImage);

}

### 基于伽马变换图像增强

伽马变换实际是对原图像的每个像素值作乘积处理，本实验中对每个像素作三次方处理。

1. **伽马变换图像增强的实现**
2. 声明函数

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

void ImgEnhanceGamma(Mat frame);

其中，参数frame为需要处理的一帧图像。

1. 函数实现

算法实现流程如下：

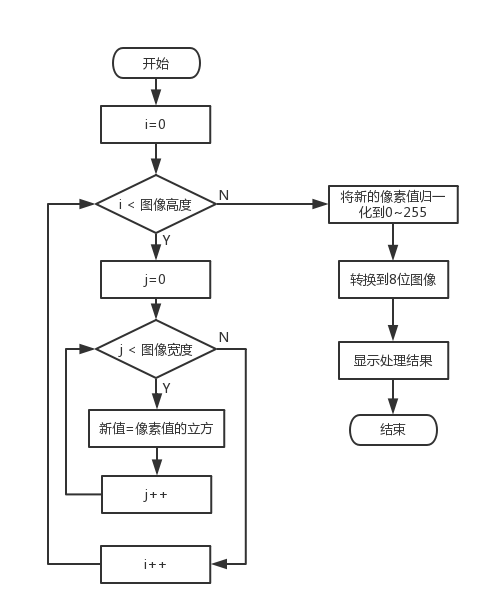


图 13

打开‘ImageHandler.cpp’文件，键入如下程序：

void ImageHandler::ImgEnhanceGamma(Mat frame)

{

Mat resultImage(frame.size(), CV\_32FC3);

for (int i = 0; i < frame.rows; i++)

{

for (int j = 0; j < frame.cols; j++)

{

resultImage.at<Vec3f>(i, j)[0] = (frame.at<Vec3b>(i, j)[0])\*(frame.at<Vec3b>(i, j)[0])\*(frame.at<Vec3b>(i, j)[0]);

resultImage.at<Vec3f>(i, j)[1] = (frame.at<Vec3b>(i, j)[1])\*(frame.at<Vec3b>(i, j)[1])\*(frame.at<Vec3b>(i, j)[1]);

resultImage.at<Vec3f>(i, j)[2] = (frame.at<Vec3b>(i, j)[2])\*(frame.at<Vec3b>(i, j)[2])\*(frame.at<Vec3b>(i, j)[2]);

}

}

//归一化到0~255

normalize(resultImage, resultImage, 0, 255, CV\_MINMAX);

//转换成8bit图像显示

convertScaleAbs(resultImage, resultImage);

imshow("伽马变换图像增强", resultImage);

}

### 六、从Kinect获取彩色图像并进行图像处理

**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：

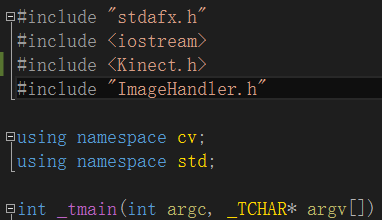


图 14

**2. 获取Kinect彩色图像**

从Kinect获取图像的一般过程为：

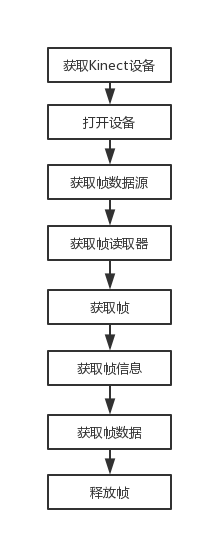


图 15

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);

// 图像增强处理

imgHandler.ImgEqualize(img);

imgHandler.ImgEnhanceGamma(img);

imgHandler.ImgEnhanceLaplace(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 16

程序成功运行后，将弹出如下窗口：

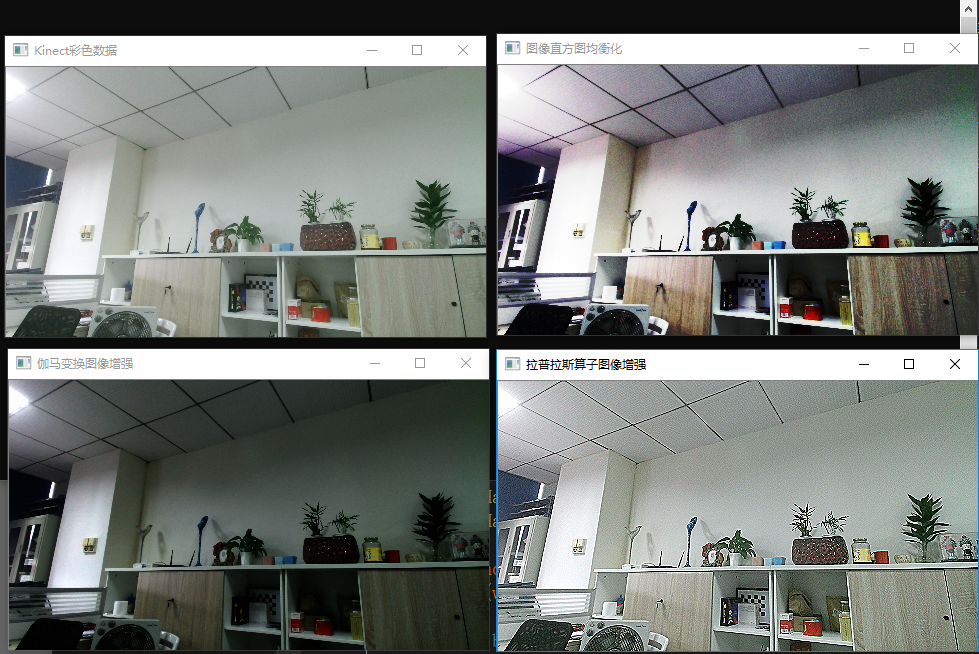


图 17

## 【思考题】

### 选择题

题目1：关于 RGB 色系下的彩色图像，下列说法正确的是：（ A ）

A、彩色图像的红色分量、绿色分量、蓝色分量都是灰度图像。

B、该彩色图像的红色分量是彩色图像。

C、若某个像素点的值是（0，255，0），则表示该颜色中只含红色。

D、若某个像素点的值是（255，255，255），则表示该颜色为黑色。

题目2：图象与灰度直方图间的对应关系是：（ B ）

A、一一对应

B、多对一

C、一对多

D、都不对

### 2、简答题

题目1： 数字图像处理的主要研究内容包含很多方面，请列出并简述其中的 4 种。

① 图像数字化：将一幅图像以数字的形式表示。主要包括采样和量化两个过 程。

② 图像增强：将一幅图像中的有用信息进行增强，同时对其无用信息进行抑 制，提高图像的可观察性。

③ 图像的几何变换：改变图像的大小或形状。

④ 图像变换：通过数学映射的方法，将空域的图像信息转换到频域、时频域 等空间上进行分析。

⑤ 图像识别与理解：通过对图像中各种不同的物体特征进行定量化描述后， 将其所期望获得的目标物进行提取， 并且对所提取的目标物进行一定的定量分析。