# 基础知识+环境搭建

## 基础知识

<https://coursehome.zhihuishu.com/courseHome/2072175#onlineCourse>

## 环境搭建

Vs2010

**vs2013+opencv2.4.10**

vs2017 opencv3

vs2019 opencv4

### vs2013安装

IE10

根据提示安装即可，需要保证足够的磁盘空间

### opencv2.4.10安装：

1）将opencv2.4.10.exe拷贝到F盘双击解压，解压后将文件夹名opencv改为opencv2410

2）环境变量配置，添加

F:\opencv2410\build\x64\vc12\bin

F:\opencv2410\build\x86\vc12\bin

### 新建vs2013项目，属性配置opencv环境

右键工程名——属性

1） VC++目录 —— 包含目录：

F:\opencv2410\build\include

F:\opencv2410\build\include\opencv

F:\opencv2410\build\include\opencv2

2） VC++目录 ——库目录：

F:\opencv2410\build\x64\vc12\lib

3） 链接器 —— 输入——附加依赖项：

opencv\_calib3d2410d.lib

opencv\_contrib2410d.lib

opencv\_core2410d.lib

opencv\_features2d2410d.lib

opencv\_flann2410d.lib

opencv\_gpu2410d.lib

opencv\_highgui2410d.lib

opencv\_imgproc2410d.lib

opencv\_legacy2410d.lib

opencv\_ml2410d.lib

opencv\_nonfree2410d.lib

opencv\_objdetect2410d.lib

opencv\_ocl2410d.lib

opencv\_photo2410d.lib

opencv\_stitching2410d.lib

opencv\_superres2410d.lib

opencv\_ts2410d.lib

opencv\_video2410d.lib

opencv\_videostab2410d.lib

## 头文件

#include <opencv2/core/core.hpp>

#include <opencv2/imgproc/imgproc.hpp>

#include <opencv2/opencv.hpp>

#include <opencv2/highgui/highgui.hpp>

#include <opencv2/video/video.hpp>

#include <opencv2/video/tracking.hpp>

#include <opencv2/video/background\_segm.hpp>

#include <opencv2/contrib/contrib.hpp>

#include <string>

#include <iostream>

## 命名空间

using namespace cv; （opencv的命名空间）

using namespace std; （C++标准库）

## 模块介绍

Core（核心组件）：基础结构及操作、动态结构、数组操作等

Imgproc（图像处理）：滤波、几何变换、结构分析、形状描述等

Highgui（GUI及视频I/O）：用户界面、读/写图像及视频、QT等

Video（视频分析）：运动检测、目标跟踪等

Objdetect（目标检测）：级联分类器及SVM

Ml（机器学习）：统计模型、贝叶斯分类器、KNN、SVM、决策树

## 环境验证 Mat类

### 图片读取、显示、保存

imread() imshow() waitKey() imwrite()

### 视频读取播放、慢放、快进

string videoFile = "xxx.mp4";

VideoCapture capture;

capture.open(videoFile);

Mat frame;

while (capture.read(frame))

{

}

### 构建自己的图像

高 .rows 宽 .cols 通道数 .channels()

1）创建一个纯色灰度图

Mat(int rows, int cols, int type, Scalar(val));

(type is CV\_8UC1, CV\_64FC3, CV\_32SC(12) etc.)

2）创建一个纯色彩色图

3）对图像像素值进行修改

### 作业

创建一个彩色背景图像，显示自己的序号

# 灰度化+二值化

## 灰度化

平均法：

加权法：

## 颜色空间

BGR YUV HSL YCrCb

## 二值化



## 练习

对图片seq\_0141.jpg进行二值化，得到背景和前景图像。

# 几何变换

## 坐标映射



f(x,y) 原图(x,y)坐标的像素值，经过映射规则T，得到目标图像g在（x’,y’）坐标的像素值。

void remap(

InputArray src, #源图像

OutputArray dst, #目标图像，与src相同大小

InputArray map1, # x坐标

InputArray map2, #y坐标

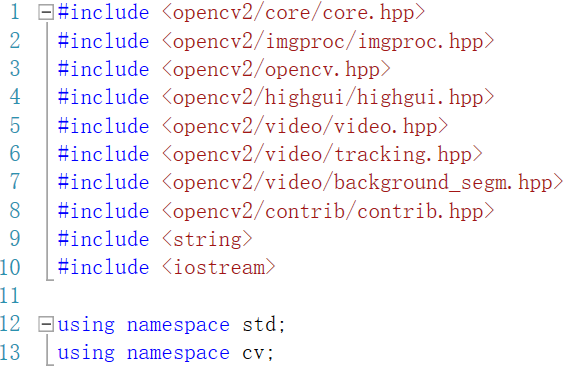
int interpolation, #表示插值方法

int borderMode=BORDER\_CONSTANT,

#表示边界插值类型

const Scalar&borderValue=Scalar()) #表示插值数值

练习：对图像做水平拉伸变换。



## 坐标平移



平移后图像缺失，图像大小不变，丢失的部分纯色

不缺失，图像变大，纯色填充

## 图像缩放

缩小到1/2 采样（深度学习里面 池化pool）

放大2倍 插值

非线性插值 u律

resize()

超分辨率成像

## 仿射变换

三组点可以求出映射关系

计算仿射变换矩阵——点

getAffineTransform

计算仿射变换矩阵——角

getRotationMatrix2D

图像仿射变换

warpAffine

## 透视变换

四个点都可以变换

getPerspectiveTransform

warpPerspective

# 直方图

## 灰度直方图

柱状图 统计频次

44个分数

56 59 87 65 78

0~60： 2

61~80：17

81~100：25

0~255

0:10

1：

2：

…

255：

直方图特征

## 彩色直方图

全局特征

split

merge

## 图像增强/图像均衡

增加对比度

equalizeHist

## 练习

原图 原图均衡后的图像

原图3个通道 对应的直方图

原图3个通道均衡后的图 对应的直方图

14个图

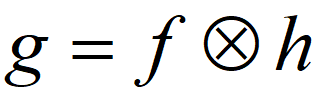
# 图像平滑

## 图像卷积

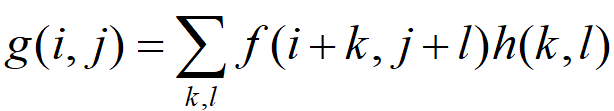
*f*：原始图像

*h*：卷积核（3\*3 5\*5 7\*7）

*g*：卷积后图像



具体到像素点(x,y)为：



举例：



0 1 2 2 3 3 4 5 255

https://www.jianshu.com/p/fc9175065d87

g 3\*3大小

传统的机器学习方法：卷积核固定（人为设定的）

深度学习：自动学习卷积核（权重），自动学习特征。

Mat myFilter2D(Mat org)

{

Mat dst(org.size(),org.type());

float kernel1[3][3] = {{0.0,1.0/8,0}, {0.0,1.0/2,0}, {0.0,1.0/8,0.0}};

Mat kern1(Size(3, 3), CV\_32FC1, kernel1);

Mat kern = (Mat\_<float>(3,3) << 0, 1, 0,

1, 0, 1,

0, 1, 0)/ (float)(4);

filter2D(org, dst, org.depth(), kern1);

return dst;

}

## 图像加噪

椒盐噪声 黑/白 0/255

随机性， 200\*200图像

比率，100% 黑白电视（雪花）

5% 2.5%的像素值被改为0，还有2.5%像素值改为255

1000个像素值被改为0，1000个像素值被改为255

用概率来改，每个像素点，有2.5%的概率改为0，有2.5%的概率改为255。

通信 信道 干扰 模拟干扰

高斯白噪声

## 图像滤波

### 均值滤波

卷积核中每个取值都一样





void blur( InputArray src, OutputArray dst, Size ksize); Size(3,3)

### 中值滤波

用掩码部分的中值表示

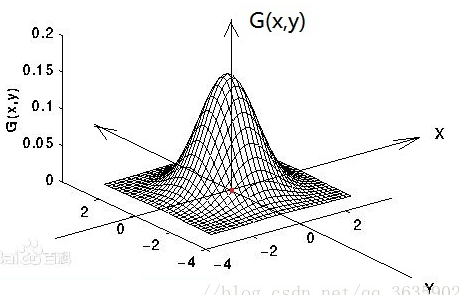


void medianBlur(InputArray src, OutputArray dst, Int ksize);

### 高斯滤波

加权的均值滤波





void GaussianBlur (InputArray src, OutputArray dst, Size ksize, double sigmaX, double sigmaY=0);

练习

(a)原始彩色图像（根据需要缩小）

(b)椒盐噪声图像，比率为5%

(c)3\*3均值滤波

(d)5\*5均值滤波

(e)3\*3中值滤波

(f)5\*5中值滤波

(g)3\*3高斯滤波

(h)5\*5高斯滤波

### 双边滤波

对低频滤波较好，高频较差。

bilateralFilter( InputArray src,

OutputArray dst,

int d,

double sigmaColor,

double sigmaSpace,

int borderType=BORDER\_DEFAULT );

参数解释：

d:每个像素邻域的直径范围。如果这个值是非正数，则函数会从第五个参数sigmaSpace计算该值。

sigmaColor: 颜色空间滤波器的sigma值，这个参数的值越大，表明该像素邻域内有越宽广的颜色会被混合到一起，产生较大的半相等颜色区域。

igmaSpace: 坐标空间滤波器的sigma值，如果该值较大，则意味着颜色相近的较远的像素将相互影响，从而使更大的区域中足够相似的颜色获取相同的颜色。

### 导向滤波

滤波同时尽可能保留边缘信息。

## 图像修复

inpaint



# 边缘检测

## 图像差分

边缘/轮廓：图像中亮度变化剧烈的像素点构成的集合。

梯度表示：微分，图像中用差分表示微分。

差分：

一阶差分：

二阶差分：

图像x方向一阶差分：



图像y方向一阶差分：



图像x方向二阶差分：



图像y方向二阶差分：



图像边缘模板

(a)水平边缘 (b)垂直边缘 (a)对角线边缘

练习：分别计算水平差分图像、垂直差分图像和边缘图像。

## Sobel Scharr 算子

Sobel算子 垂直边缘 水平边缘

scharr算子  

addWeighted

## Laplacian 拉普拉斯/傅里叶

Laplacian

## Canny

1）噪声消除：高斯滤波

2）计算梯度幅度和方向：用sobel模板

3）非极大值抑制（去掉伪边缘）：局部像素和梯度最大

4）滞后阈值求解边缘：梯度幅度高于大阈值保留；小于小阈值排除；二者之间时，邻近像素阈值高于大阈值时保留。

void Canny( InputArray image, OutputArray edges,

double threshold1, // 值越大，边缘越少

double threshold2, // 推荐值为threshold1的3倍

int apertureSize=3, bool L2gradient=false );

原始图像

Sobel\_3\*3\_垂直边缘图像

Sobel\_3\*3\_水平边缘图像

Sobel\_3\*3\_水平图像

Scharr\_3\*3\_垂直边缘图像

Scharr\_3\*3\_水平边缘图像

Scharr\_3\*3\_边缘图像

Laplacian \_3\*3\_边缘图像

Canny\_threshold1\_50\_边缘图像

Canny\_threshold1\_100\_边缘图像

## 直线检测

HoughLinesP

# 综合练习1

基于车辆直线检测的车道排队拥堵分析

# 运动检测

## 基本知识

视频暂停 ROI 画线 画矩形框 添加文本

## 运动前景

背景 前景

BackgroundSubtractorMOG2 mog;

## 形态学处理

<https://blog.csdn.net/zhangjunp3/article/details/79831117>

Mat element = getStructuringElement(MORPH\_RECT, Size(3, 3));

腐蚀：erode 膨胀：dilate

## 轮廓检测

void findContours( InputOutputArray image, OutputArrayOfArrays contours, OutputArray hierarchy, int mode, int method, Point offset=Point());

vector<vector<Point>>contours;

vector<Vec4i>hierarchy;

mode轮廓检索模式：

RETR\_EXTERNAL:表示只检测最外层轮廓，对所有轮廓设置hierarchy[i][2]=hierarchy[i][3]=-1

RETR\_LIST:提取所有轮廓，并放置在list中，检测的轮廓不建立等级关系

method:轮廓近似方法

CHAIN\_APPROX\_NONE：获取每个轮廓的每个像素，相邻的两个点的像素位置差不超过1

CHAIN\_APPROX\_SIMPLE：压缩水平方向，垂直方向，对角线方向的元素，值保留该方向的终点坐标，例如一个矩形轮廓只需4个点来保存轮廓信息

Rect r = boundingRect(contours[i]);

# 综合练习2

制高点车流量统计分析

# 图像特征提取

传统的图像识别：

adaboost(级联分类器) + 特征提取 + SVM(支持向量机)二分类

特征提取 颜色直方图、灰度直方图、纹理特征LBP、梯度特征HOG、转移概率

深度学习：神经网络提取特征 + SVM分类

端-端

SVM进行分类：

数据集：训练集（训练分类模型）+ 测试集（测试模型准确率）

数据预处理，归一化，正负样本图片归一化到相同大小

## 纹理特征LBP



## 梯度特征HOG

1）灰度化，归一化到100\*100

2）计算每个像素的梯度（包括大小和方向）,98\*98



的取值范围？

3）将梯度图像划分为14\*14个小单元（7\*7）

4）计算每个小单元的梯度直方图（维度为9）。说明1：小单元中某个值的介于20~40度，则直方图第二个值计数+=梯度大小。

5）将2\*2个小单元的直方图串联成一个大直方图（36维），并进行归一化，36个数除以最大的，[0,1]。记2\*2个小单元为一个大单元，则总共有多少个大单元？

6）将所有大单元的特征合并成一个特征，最终的特征维度为多少？

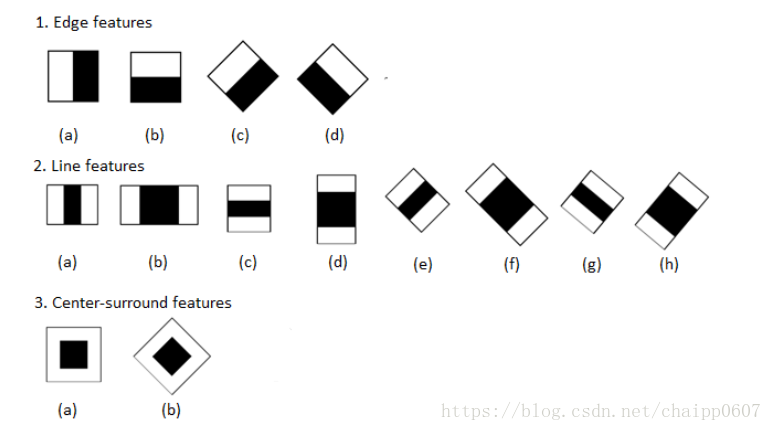
HOGDescriptor hog(Size(88, 24), Size(16, 16), Size(8, 8), Size(8, 8), 9);//HOG检测器

vector<float> descriptors;//HOG描述子向量

hog.compute(grayImg, descriptors);

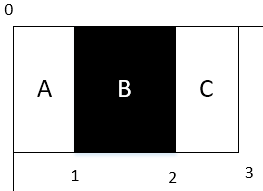
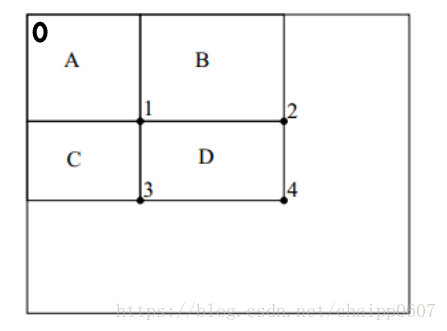
## 目标检测Haar

1） haar-like特征：矩形模板中白色矩形内所有像素值的和减去黑色矩形内所有像素值的和。

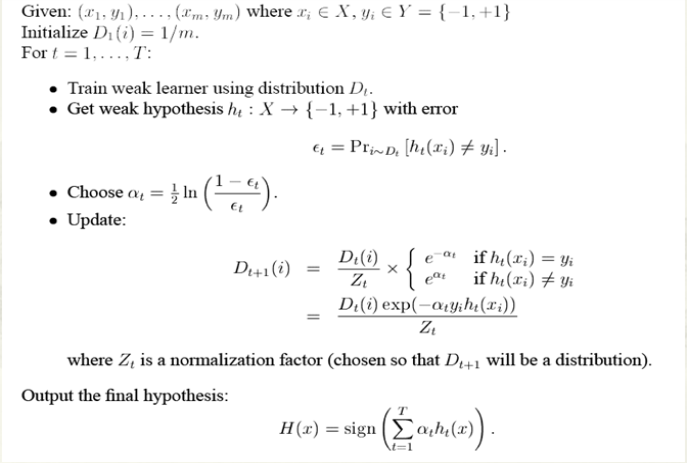


3种类型4种形式模板

2）特征快速计算——积分图

3）级联分类器训练（adaboost） 强分类器 = 若干个弱分类器级联



https://blog.csdn.net/haidao2009/article/details/7514787

# 综合练习3

基于联合多特征的人脸检测与匹配

# 综合练习4

道路车辆违法检测系统