深圳大学 SHENZHEN UNIVERSITY

计算机视觉

南区计软328

群名称:计算机视觉2024春
群　号:731395587

深圳大学
计算机与软件学院

实验内容

序号	实验主题	实验内容	实验要求	实验时数	每组人数	实验类型
1	图像处理应用实验	1．熟悉图像的表示及基本元素、通道操作； 2．掌握基本图像增强方法； 3．掌握OpenCV计算机视觉库；	必做	6	1	讲授+实验
2	图像特征提取及综合应用实践	1．熟悉图像处理基本操作； 2．掌握图像边缘检测原理； 3．掌握图像基本特征抽取以及在实际问题中的应用；	必做	6	1	讲授+实验
3	计算机视觉系统实践	1．熟悉计算机视觉分类任务； 2．掌握数据集的准备及模型训练过程； 3．培养应用计算机视觉解决问题的能力；	必做	6	1	讲授+实验

涉及学科

- 数字图像处理
- 计算机视觉
- 模式识别
- 程序设计

实验考察形式

- 三次实验报告（在Blackboard发布与提交）
- 平时编程练习及表现
- 实验汇报

实验考察内容

- 计算机视觉与模式识别基础知识掌握能力
- 算法设计能力
- 编程及系统架构能力
- 论文写作及表达能力

实验一

实验一：图像增强应用实践

实验目的：

1. 熟悉图像的表示及基本元素、通道操作；
2. 掌握基本图像增强方法；
3. 掌握OpenCV计算机视觉库；

实验要求：

1. 实验提交文件为实验报告和相关程序代码，以压缩包的形式提交，实验报告命名规则为“计算机视觉-学号-姓名-实验报告1.doc”，其他文件打包成压缩文件，命名为“计算机视觉-学号-姓名-实验报告1-其他.zip”；
2. 所有素材和参考材料需列明出处，实验报告中的图片和程序代码建议标注个人水印或标识信息：姓名，班级，学号信息；

二、实验内容：

不调用库函数，自己动手编程实现图像增强相关方法，并与OpenCV的库函数进行效果对比分析；

三、实验时间

实验报告统一命名为：计算机视觉-学号-姓名-实验报告1.doc

如有提交其他文件或程序，统一命名为：计算机视觉-学号-姓名-实验报告1-其他.zip

实验时间：2024年3月4日至2024年4月13日（注意按时提交，不要延期）

实验报告提交时间：2024年4月13日下午5:00

实验一

计算机视觉-学号-姓名-实验报告1.doc [兼容模式] - Word

一、实验目的与要求

实验目的：

1. 熟悉图像的表示及基本元素、通道操作；
2. 掌握基本图像增强方法；
3. 掌握 OpenCV 计算机视觉库；

实验要求：

1. 实验提交文件为实验报告和相关程序代码，以压缩包的形式提交，实验报告命名规则为“计算机视觉-学号-姓名-实验报告 1.doc”，其他文件打包成压缩文件，命名为“计算机视觉-学号-姓名-实验报告 1-其他.zip”；

2. 所有素材和参考材料需列明出处，实验报告中的图片和程序代码建议标注个人水印或标识信息：姓名，班级，学号信息；

二、实验内容与方法

实验内容：不调用库函数，自己动手编程实现图像增强相关方法，并与 OpenCV 的库函数进行效果对比分析；

三、实验步骤与过程

提示（提交作业时请删除红底标注提示信息）：根据实验内容自己组织填写，内容截图插入实验报告后大小要调得适中，并且必须加上相关的说明，每页图片建议不超过 6 张，排版的美观也加入评分标准。（以后所有的实验报告都是这个要求，不再重复说明）。

四、实验结论或体会

提示（提交作业时请删除红底标注提示信息）：有重要结论与心得体会，包括技术上的或设计上的、操作上的，理论上的等等。

五、思考题

对比不同图像增强方法的优势、劣势，包括其合适的应用场景。

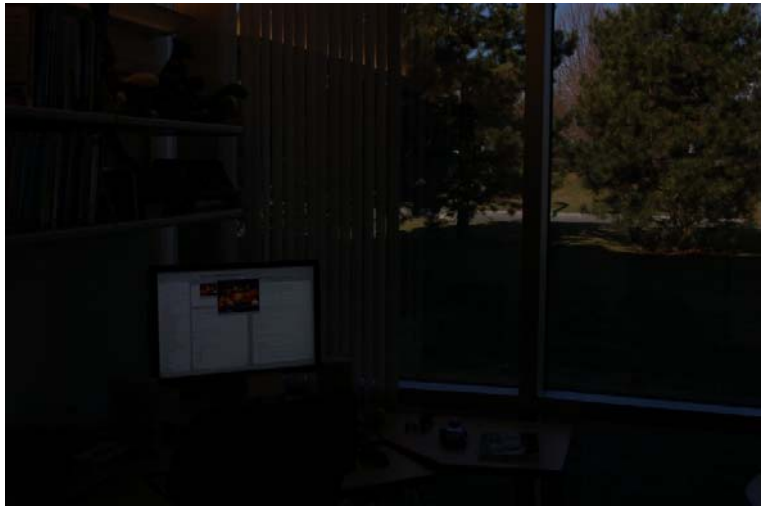
实验一

图像增强方法：

- 基于滤波的图像增强方法——局部二值模式、中值滤波、均值滤波、高斯滤波、图像锐化、Gabor滤波；
- 基于统计、函数映射的图像增强方法——直方图均衡化、直方图规定化、灰度变换、伽马变换；
- 其他图像增强方法——基于字典学习、稀疏表示、深度学习的方法。

- 实验一

car



office

foggyInput

coins

实验一

cameraman

高斯噪声 均值0方差0.005

均值0方差0.01

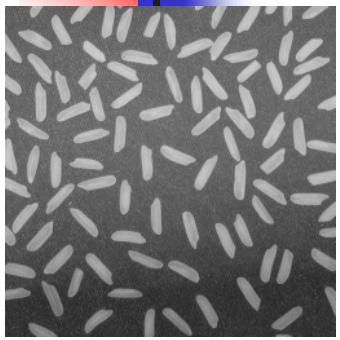
均值0方差0.015

椒盐噪声 噪声密度0.1

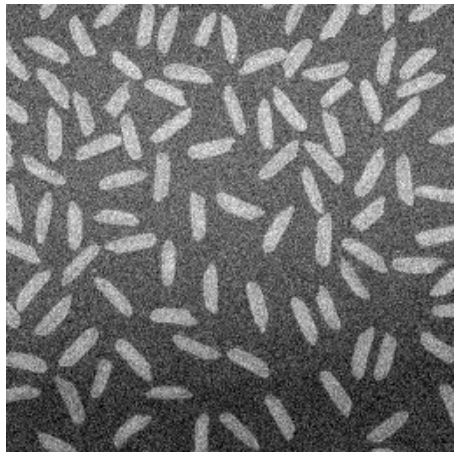
噪声密度0.2

噪声密度0.3

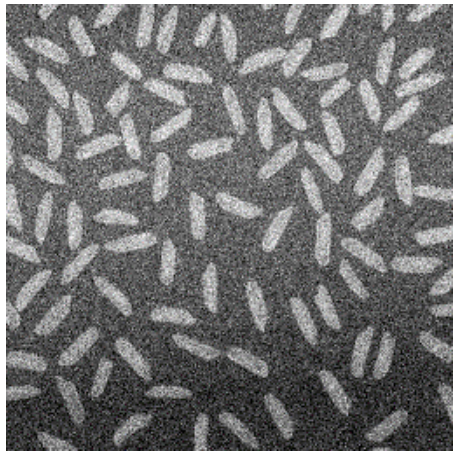
实验一



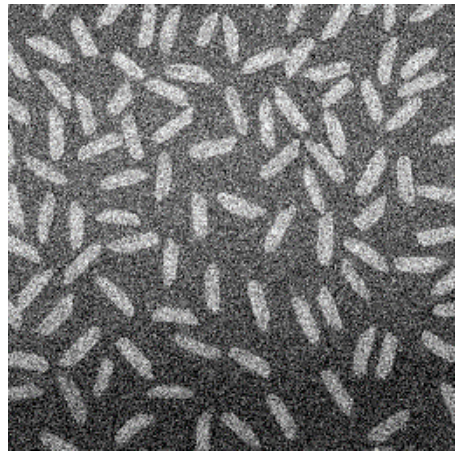
rice



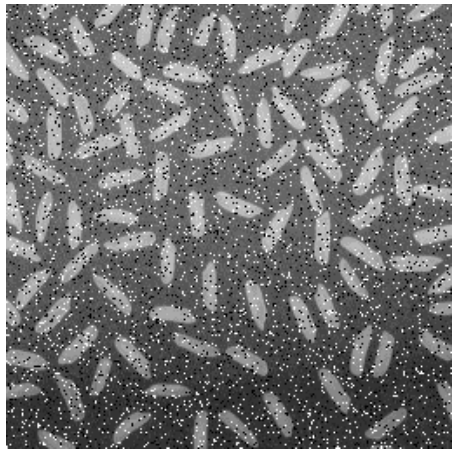
高斯噪声 均值0方差0.005



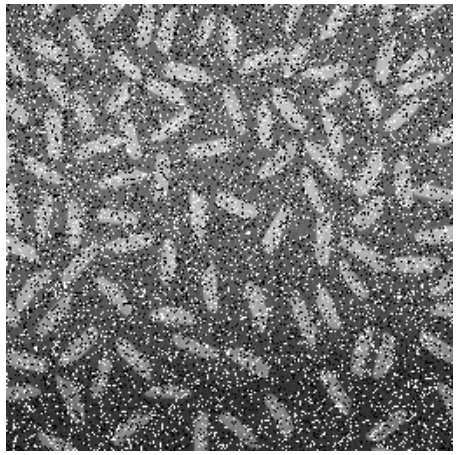
均值0方差0.01



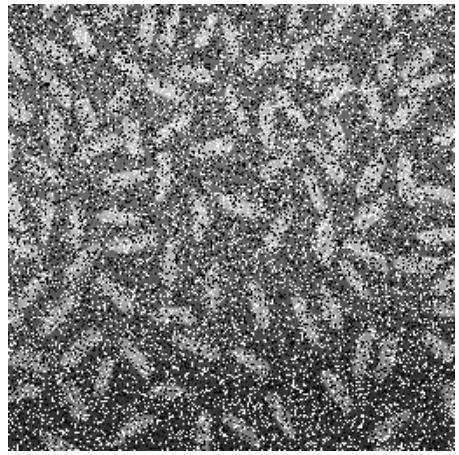
均值0方差0.015



椒盐噪声 噪声密度0.1

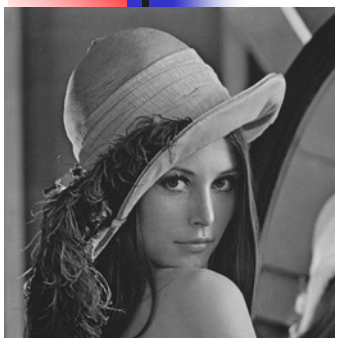


噪声密度0.2



噪声密度0.3

实验一



lena

高斯噪声 均值0方差0.005

均值0方差0.01

均值0方差0.015

椒盐噪声 噪声密度0.1

噪声密度0.2

噪声密度0.3

- 实验一

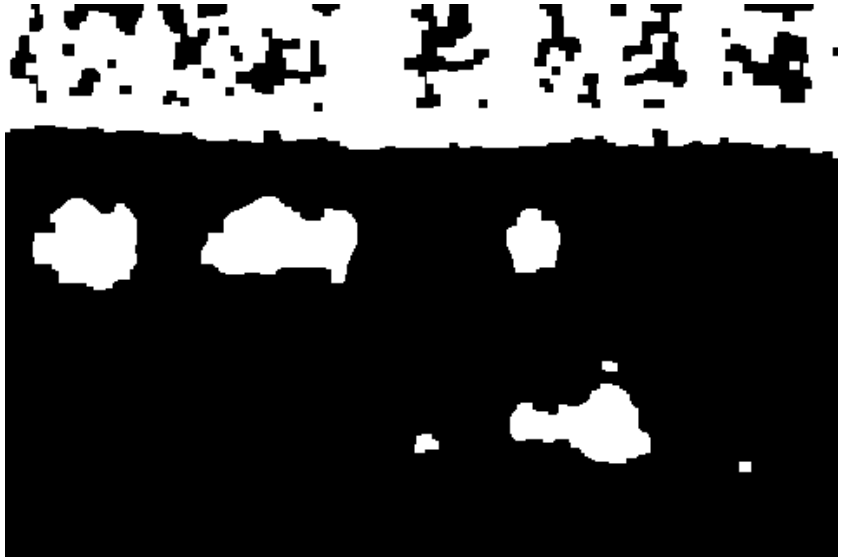
coloredChips

peppers

- 实验一

boldt

实验一



group

Group_binary

OpenCV中关于图像操作

获取二值图像轮廓

findContours(InputOutputArray image, OutputArrayOfArrays contours, OutputArray hierarchy, int mode, int method, Point offset=Point());
第一个参数：image表示输入的原图像，是一个单通道的二值图像

第二个参数：contours表示输出的轮廓集合，是vector<vector<Point>>类型的，每一个轮廓都用一系列的像素点point（col,row）表示出来

第三个参数：hierarchy表示返回的各个轮廓的拓扑结构，就是上面那个树形结构，遍历contours的时候就是靠着这个树形结构实现的，相当于先序遍历这个树，因此如果一个轮廓有内嵌轮廓，那么在遍历过当前轮廓之后，下一个遍历的轮廓一定是当前轮廓的内嵌轮廓，这也是二维码定位用这个函数实现的基础。数据格式vector<vec4i>,0~4分别存储后一个轮廓，前一个轮廓，父轮廓和内嵌轮廓的索引号，如果不存在就是-1。

- 实验一

Road

- OpenCV中关于图像操作

图像缩放

void cv::**resize** (InputArray src, OutputArray dst, Size dsize, double fx = 0, double fy = 0, int interpolation =INTER_LINEAR)

图像镜像

void cv::**flip**(InputArray src, OutputArray dst, int flipCode)

$$\texttt{dst}_{ij} = \begin{cases} \texttt{src}_{\texttt{src.rows}-i-1,j} & if\ \texttt{flipCode} = 0 \\ \texttt{src}_{i,\texttt{src.cols}-j-1} & if\ \texttt{flipCode} > 0 \\ \texttt{src}_{\texttt{src.rows}-i-1,\texttt{src.cols}-j-1} & if\ \texttt{flipCode} < 0 \end{cases}$$

仿射变换（目标校正）

cv::warpAffine()

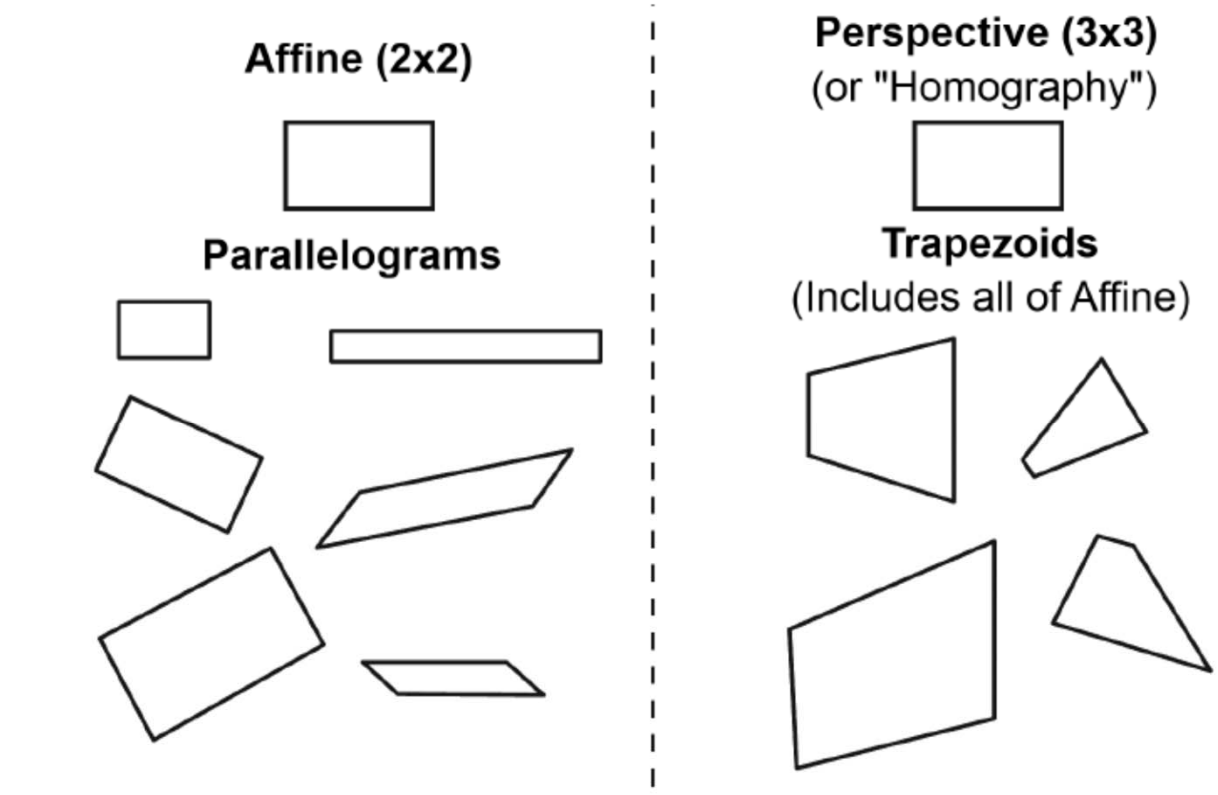


Figure 11-3. Affine and perspective transformations

OpenCV中关于图像操作

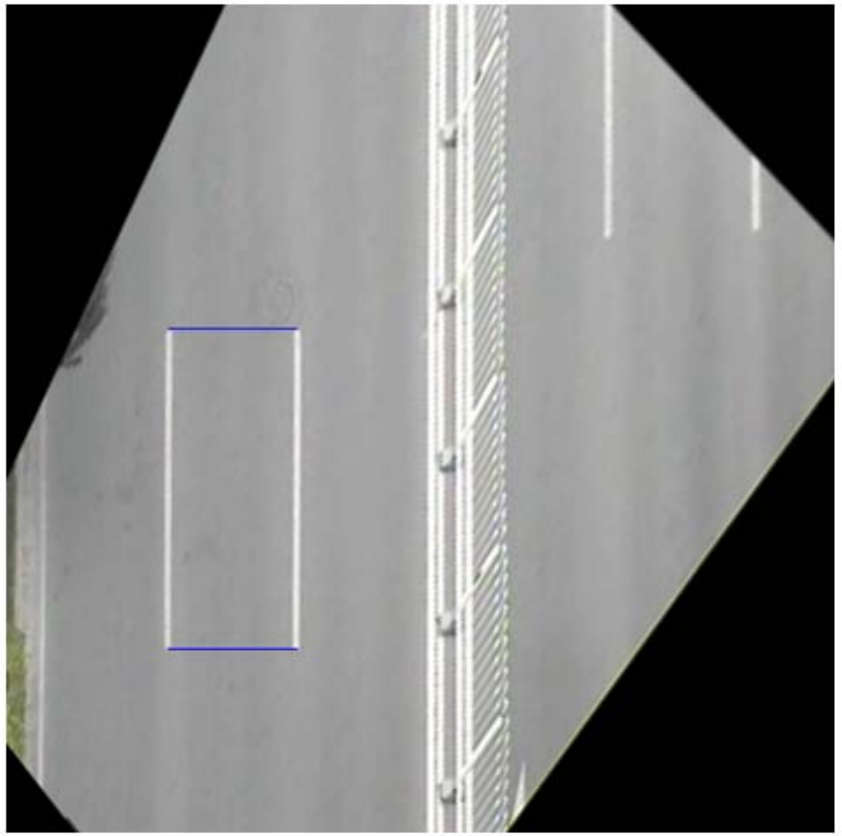
仿射变换

```
void cv::warpAffine ( InputArray  src, OutputArray  dst, InputArray  M, Size
dsize, int  flags = INTER_LINEAR, int  borderMode = BORDER_CONSTANT, const
Scalar &  borderValue = Scalar() )
```

旋转矩阵

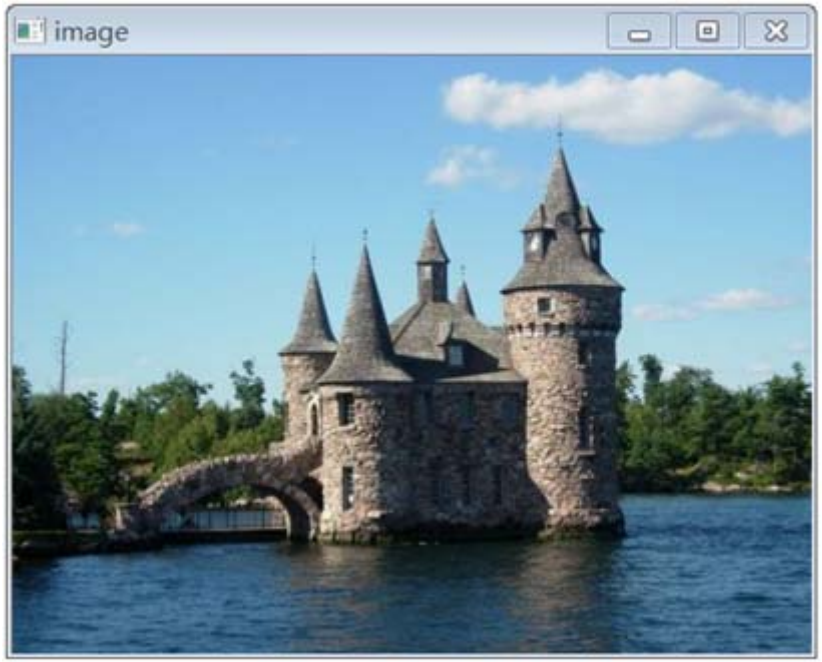
```
Mat cv::getRotationMatrix2D (Point2f center, double angle, double scale)
Ex1:
cv::Mat rot_mat = cv::getRotationMatrix2D(center, rotate_degree, 1.0);
cv::warpAffine(img_src, img_rotate, rot_mat, src_sz);
Ex2:
cv::Rect2f bbox = cv::RotatedRect(cv::Point2f(), img_src.size(),
rotate_degree).boundingRect();
// adjust transformation matrix--translation transformation
rot_mat.at<double>(0, 2) += bbox.width / 2.0 - img_src.cols / 2.0;
rot_mat.at<double>(1, 2) += bbox.height / 2.0 - img_src.rows / 2.0;
cv::warpAffine(img_src, img_urotate, rot_mat, bbox.size());
```

- OpenCV中关于图像操作

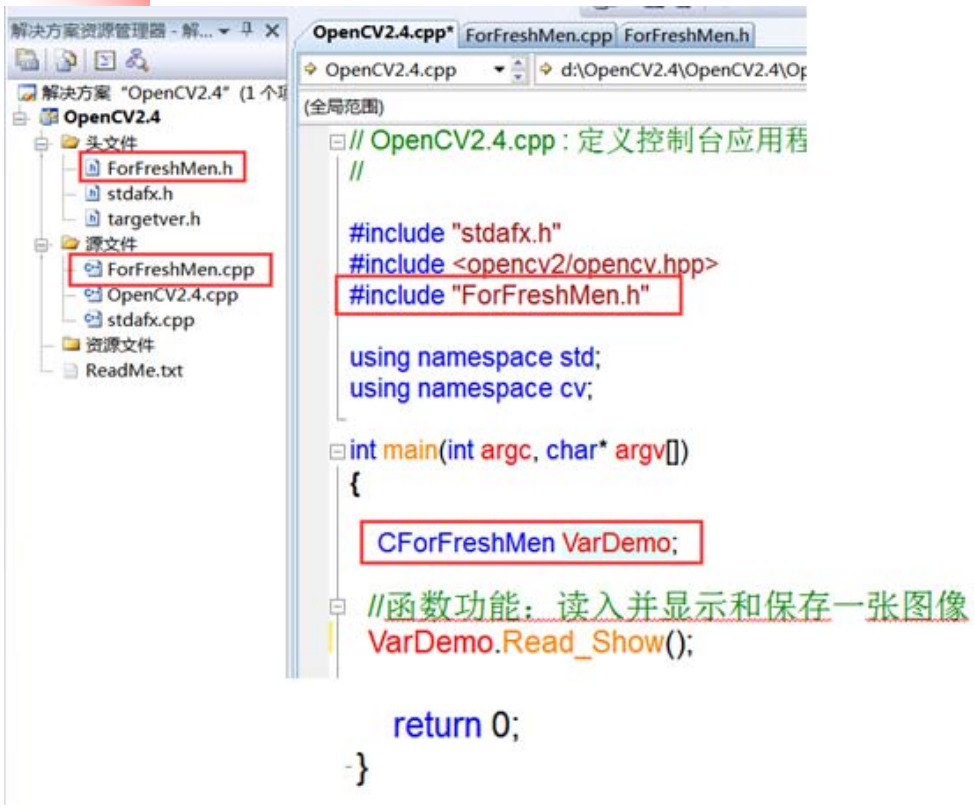


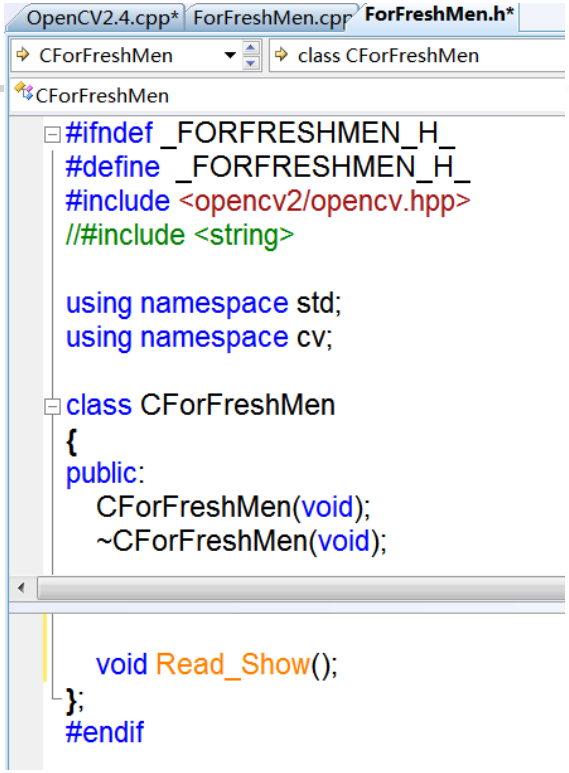
OpenCV图像读取示例

```
#include "stdafx.h"
#include <opencv2/opencv.hpp>
using namespace std;
using namespace cv;

void Read_Show();
int _tmain(int argc, _TCHAR* argv[])
{
    Read_Show();
    return 0;
}

void Read_Show()
{
    const char* imagename = "boldt.jpg";
    //从文件中读入图像
    Mat img =imread(imagename);
    //如果读入图像失败
    if(img.empty())
    {
        fprintf(stderr, "Can not load image %s\n", imagename);
        exit(0);
    }
    //显示图像
    namedWindow("image",1);
    cout<<"函数功能：读入并显示和保存一张图像"<<endl;
    imwrite("save.jpg",img);
    imshow("image", img);
    //此函数等待按键，按键盘任意键就返回
    waitKey(0);
}
```



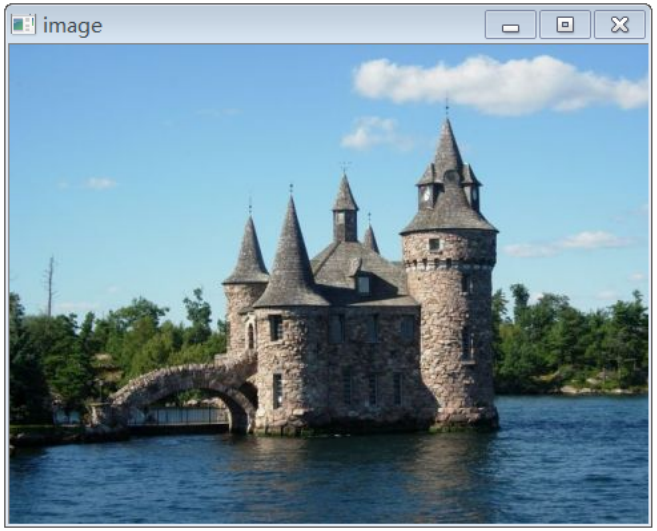
OpenCV下采用类+结构体的框架进行编程



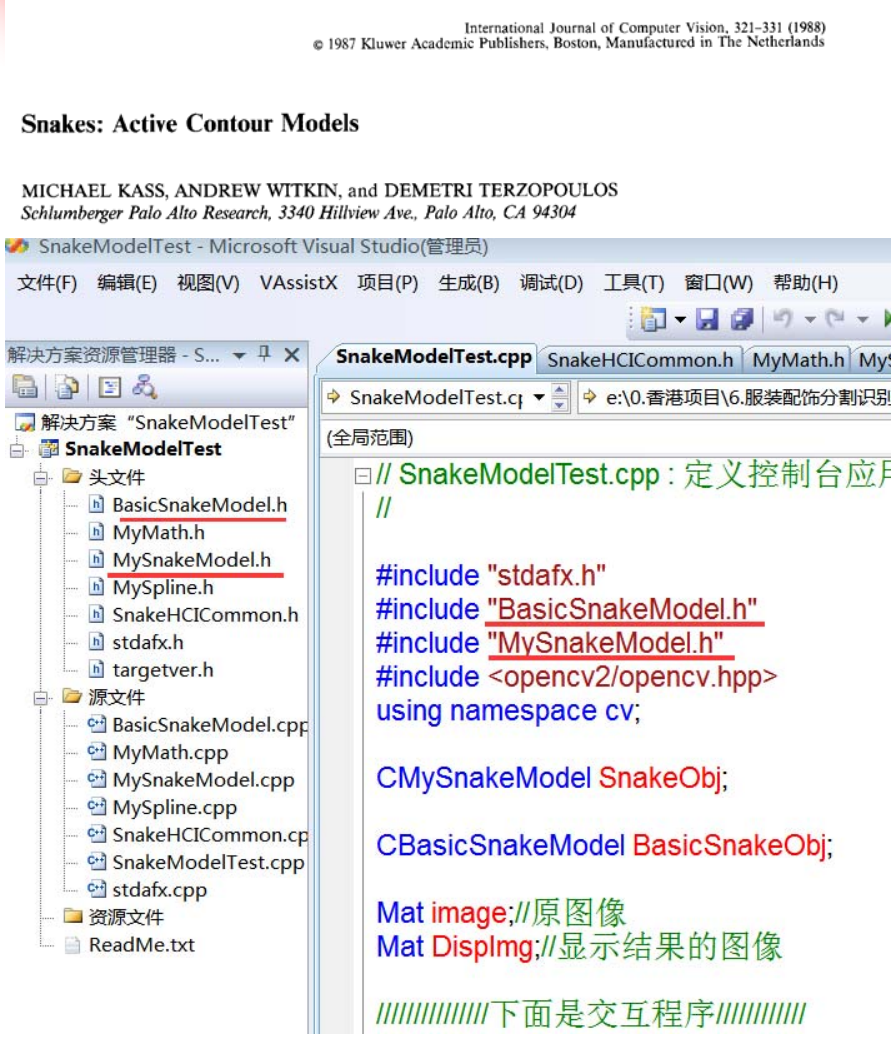


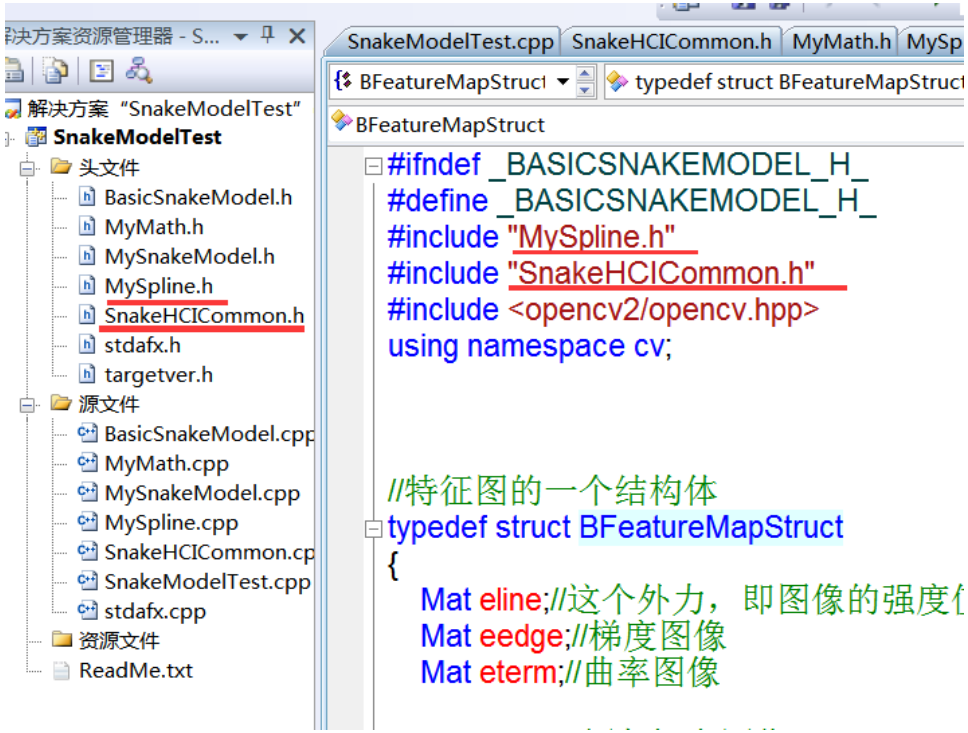
OpenCV下采用类+结构体的框架进行编程

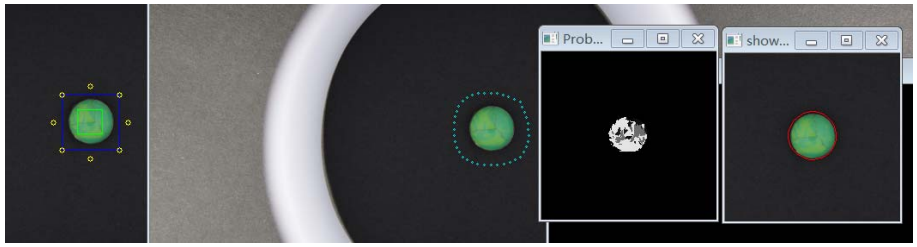
```
void CForFreshMen::Read_Show()
{
    const char* imagename = "boldt.jpg";
    //从文件中读入图像
    Mat img =imread(imagename);
    //如果读入图像失败
    if(img.empty())
    {
        fprintf(stderr, "Can not load image %s\n", imagename);
        exit(0);
    }
    //显示图像
    namedWindow("image",1);
    cout<<"函数功能：读入并显示和保存一张图像"<<endl;
    imwrite("save.jpg",img);
    imshow("image", img);
    //此函数等待按键，按键盘任意键就返回
    waitKey(0);
}
```



OpenCV下采用类+结构体的框架进行编程







一个蛇模型算法框架实例

遍历图像上所有像素（例如把图像左右翻转）

访问图像中像素的几种方法

1.使用Mat_模板子类访问图像元素

```
Mat_<uchar> img=Mat::zeros(cvSize(300,200),CV_8U);
for(int j=0;j<img.rows/2;j++)
{
   for(int i=0;i<img.cols/2;i++)
   {
      img(j,i)=255;
   }
}
cout<<"//函数功能：使用Mat_模板子类更简洁访问图
imshow("show",img);
cvWaitKey(0);
```

2.以.at<>方式访问

```
cout<<"函数功能：给图片添加椒盐噪声"<<endl;
for(int k=0;k<n;k++)
{
   int i=rand()%image.cols;
   int j=rand()%image.rows;
   if(image.channels()==1)//灰度图
   {
      image.at<uchar>(j,i)=255;
   }
   else if(image.channels()==3)//彩色图
   {
      image.at<Vec3b>(j,i)[0]=255;
      image.at<Vec3b>(j,i)[1]=255;
      image.at<Vec3b>(j,i)[2]=255;
   }
}
```

3.通过访问图像每行以及以in-place的方式访问

```
int nl=image.rows;//行数
//每行的元素个数
int nc=image.cols*image.channels();
int k=0;
cout<<"函数功能：通过访问图像每行
for(int j=0;j<nl;j++)
{
   //得到第j行的首地址
   uchar* data=image.ptr<uchar>(j);
   for(int i=0;i<nc;i++)
   {
      if(k==0)
      {
         int tt=data[i];
         cout<<tt<<endl;
      }
      //处理每一个像素
      //data[i]=data[i]/div*div+div/2;//慢速，
      *data++=*data/div*div+div/2;//快速

      if(k==0)
      {
         int tt=data[i];
         cout<<tt<<endl;
         k++;
      }
   }
}
```

4.通过迭代器方式访问

```
//得到初始位置的迭代器
Mat_<Vec3b>::iterator it=image.begin<Vec3b>();
//得到终止位置的迭代器
Mat_<Vec3b>::iterator itend=image.end<Vec3b>();
cout<<"函数功能：通过访问图像每行，以迭代器
//遍历所有像素
for(;it!=itend;++it)
{
   //处理每个像素
   (*it)[0]=(*it)[0]/div*div+div/2;
   (*it)[1]=(*it)[1]/div*div+div/2;
   (*it)[2]=(*it)[2]/div*div+div/2;
}
```

5.通过三行指针联合访问

```
//如有必要则分配图像
cout<<"函数功能：锐化图像-通过三行指针"<<endl;
result.create(img.size(),img.type());
for(int j=1;j<img.rows-1;j++)//处理除了第一行和最后一行之外的所有行
{
   const uchar*previous=img.ptr<const uchar>(j-1);//上一行
   const uchar *current=img.ptr<const uchar>(j);//当前行
   const uchar *next=img.ptr<const uchar>(j+1);//下一行
   uchar *output=result.ptr<uchar>(j);//输出行

   for(int i=1;i<img.cols-1;i++)
   {
      *output++=saturate_cast<uchar>(5*current[i]-current[i-1]-current[i+1]-previous[i]-next[i]);
   }
}
//将未处理的像素设置为0
result.row(0).setTo(Scalar(0));
result.row(result.rows-1).setTo(Scalar(0));
result.col(0).setTo(Scalar(0));
result.col(result.cols-1).setTo(Scalar(0));
```

Vector中的删除操作

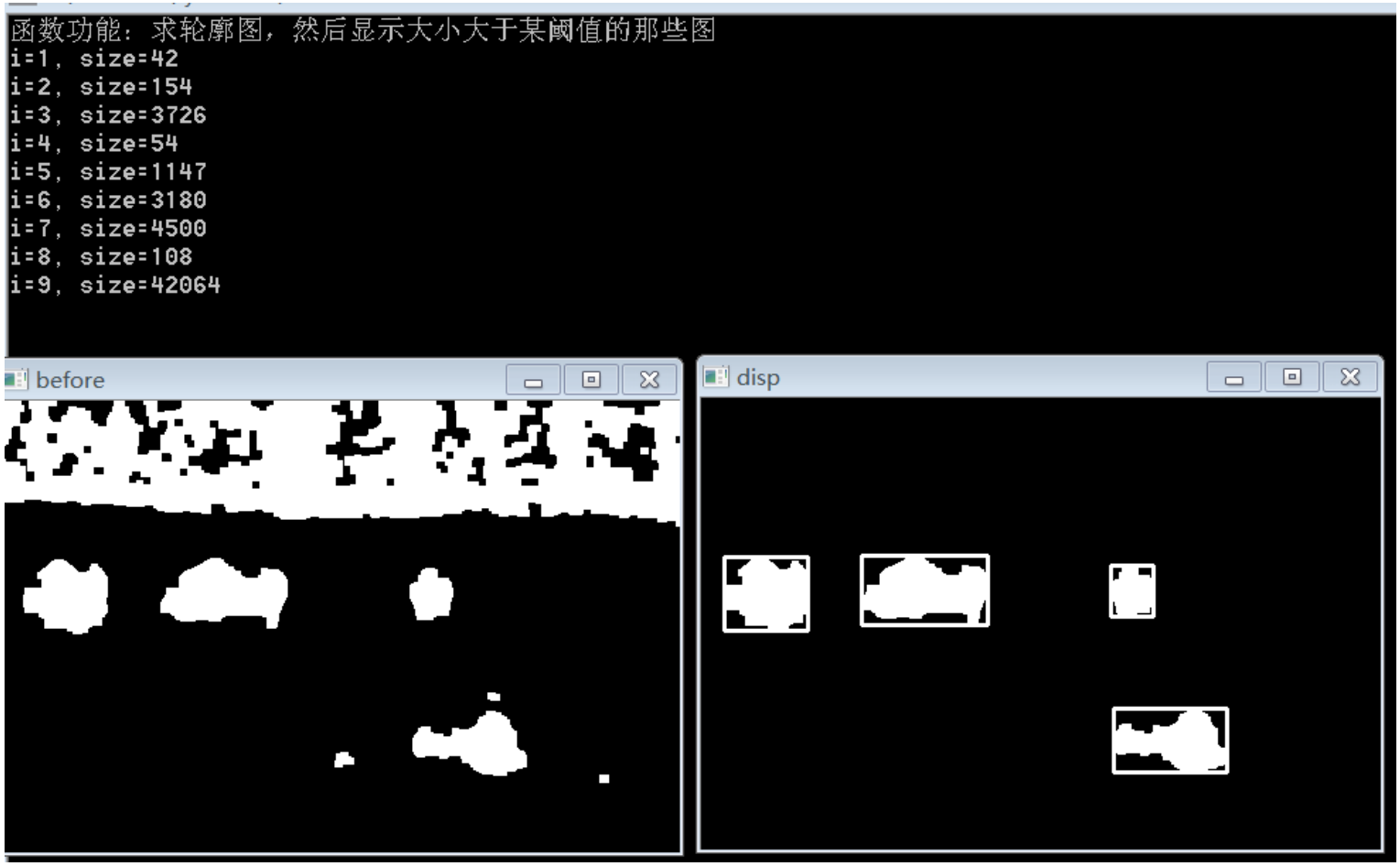
```
#include <list>
#include <vector>
using namespace std;

int main(int argc, char* argv[])
{
    //删除指定vector元素
    vector<int>intervectorlist;
    for(int i=0;i<10;i++)
    {
        intervectorlist.push_back(i);
    }
    vector<int>::iterator it_vector;
    for(it_vector=intervectorlist.begin();it_vector!=intervectorlist.end();)
    {
        if(*it_vector==2||*it_vector==7)
        {
            printf("Delete=%d\n",(*it_vector));
            it_vector=intervectorlist.erase(it_vector); //删除元素，返回值指向已删除元素的下一个位置
            printf("删除结束\n");
        }
        else
        {
            it_vector++;
        }
    }
    //输出
    printf("删除vector\n");
    for(int i=0;i<intervectorlist.size();i++)
    {
        printf("%d\n",intervectorlist[i]);
    }
    printf("删除全部\n");
    vector<int>().swap(intervectorlist);
    for(int i=0;i<intervectorlist.size();i++)
    {
        printf("%d\n",intervectorlist[i]);
    }

    printf("\n");
```

List中的删除操作

```
//删除指定list元素
list<int>interlist;
for(int i=0;i<10;i++)
{
    interlist.push_back(i);
}
list<int>::iterator it;
for(it=interlist.begin();it!=interlist.end();)
{
    if(*it==2||*it==7)
    {
        printf("Delete=%d\n",(*it));
        interlist.erase(it++); //删除元素，返回值指向已删除元素的下一个位置
        printf("删除结束\n");
    }
    else
    {
        it++;
    }
}
printf("删除全部\n");
if(interlist.size()>0)
{
    list<int>::iterator it;
    for(it=interlist.begin();it!=interlist.end();)
    {
        it=interlist.erase(it++);
    }
}
```

OpenCV获取图像中的轮廓，对图像中的目标进行计数

OpenCV获取图像中的轮廓，对图像中的目标进行计数



```
void CForFreshMen::ContourFunSomeSizeShow()
{
    cout<<"函数功能：求轮廓图，然后显示大小大于某阈值的那些图"<<endl;
    Mat image=imread("binaryGroup.bmp",0);
    Mat ShowImg (image.size(),CV_8U,Scalar(0));

    vector<vector<Point>>contours;
    imshow("before",image);
    findContours(image,    //注意，这个函数会修改image图像的内容
        contours, // a vector of contours
        CV_RETR_EXTERNAL,  //retrieve the external contours    或者：CV_RETR_LIST //retrieve all contours
        CV_CHAIN_APPROX_NONE);  // all pixels of each contours

    vector<vector<Point>>contours_tmp;
    float T=1000;//矩形面积
    for(int i=0;i<contours.size();i++)
    {
        vector<Point> Points=contours[i];//如何获取一个包围框的点集
        Rect rect=boundingRect(Points);//如何获取该点集的外围框
        printf("i=%d, size=%d\n",i+1,rect.width*rect.height);
        if(rect.width*rect.height>T&&rect.width*rect.height<5000)
        {
            //存储
            contours_tmp.push_back(Points);

            //显示
            //显示指定的轮廓
            //方法1：使用离散点的方式画出目标的轮廓
            int i;
            for(i=0;i<Points.size()-1;i++)
            {
                line(ShowImg,Points[i],Points[i+1],Scalar(255),1,8,0);
            }
            line(ShowImg,Points[i],Points[Points.size()-1],Scalar(255),1,8,0);

            //方法2：可以按照以下格式，逐个填充轮廓
            vector<vector<Point>>M;
            M.push_back(Points);
            fillPoly(ShowImg,M,Scalar(255));//画出填充结果

            //方法3：
            Rect b=boundingRect(Points);//如何获取该点集的外围框
            rectangle(ShowImg,b,Scalar(255,255,255),2);//画出框框
        }
    }
    imshow("disp",ShowImg);
    cvWaitKey(0);
}
```

上机练习内容：

1. OpenCV环境配置；
2. 实现对图像的读取；
3. 把一副彩色图像的三个通道变成3个单通道图像存储到硬盘上并显示；
4. 计算一幅单通道图像的直方图；
5. 编程实现对一幅单通道图像的边缘检测。
6. 对一幅灰度图像进行直方图均衡化
7.对图像进行二值化操作；
8. 图像形态学操作；

上机练习内容：

9. 获取图像轮廓；

10. 实现目标的计数；

11. 目标的旋转；