山东大学 计算机科学与技术 学院

计算机视觉 课程实验报告

学号：201800130086	姓名： 徐鹏博	
实验题目：图像代数运算		

实验过程中遇到和解决的问题：

（记录实验过程中遇到的问题，以及解决过程和实验结果。可以适当配以关键代码辅助说明，但不要大段贴代码。）

1. 对比度调整

设计 sigmoid 函数：

$$sig(x) = \frac{1}{1+e^{-x}}, sig \in (0,1)$$

$$sig(x) = \frac{255}{1+e^{\frac{127.5-x}{\rho}}}, sig \in (0,255)$$

ρ为参数

```
uchar sigmoid(uchar x,float e){//sigmoid 函数
        float z=(float) x; uchar y;
        z=255.0/(1+exp((127.5-z)/e));
        y=(uchar) z; return y;
}
void Contrast_Adjustment(const Mat &In, Mat &Out, float e){ //E2-1
     int r=In.rows,c=In.cols, inStep=In.step, inChannel=In.channels(),outStep=Out.step;
     uchar* InM = In.data;     uchar* OutM = Out.data;
     for(int row=0;row<r;row++){
          for(int col=0;col<c;col++){
               for(int i=0;i<inChannel;++i){
                    OutM[row * outStep + col * inChannel + i]=sigmoid(InM[row * inStep + col * inChannel +
i],e);
               }
          }
     }
     imshow(WindowName, Out);
}
```

Slider 控件需要设置创建窗口和滑条，另外需要设置回调函数。

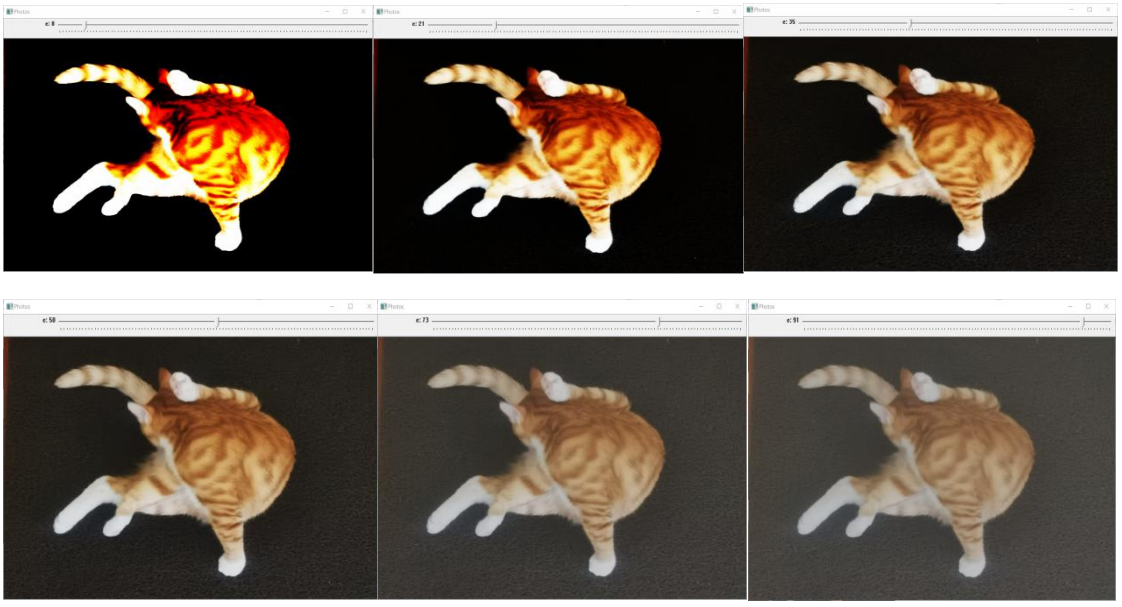
```
void on_Trackbar1(int,void*)
{
     if (Value < 1)
     {
          Value = 1; setTrackbarPos(TrackbarName, WindowName, Value);
     }
     Mat element = getStructuringElement(MORPH_RECT, Size(Value, Value));
```

```
        img1.copyTo(img2);
        Contrast_Adjustment(img1, img2, (float)Value);
}

void playWindow1(){
        Value=10;
        namedWindow(WindowName, 0);
        sprintf(TrackbarName, "e");
        createTrackbar(TrackbarName, WindowName, &Value, MaxSize, on_Trackbar1);
        on_Trackbar1(0, 0);
}
```

原图

效果:参数 ρ分别=8，21，35，50，73，91



2. 背景相减

思路:

设带前景的图 I(x, y)，背景图 B(x, y)，输出 mask=f(x, y)，d 为 I 和 B 的欧式距离，p 为设定阈值。距离大于等于阈值取 255(1)， 小于阈值取 0(0)。

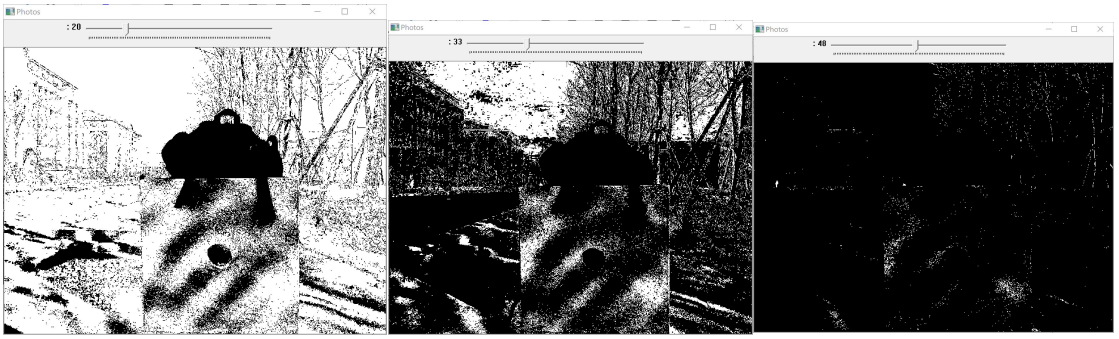
$$d = \| I(x,y|r,g,b) - B(x,y|r,g,b \|_2$$

$$f(x,y) = \begin{cases} 255 & d \geq p \\ 0 & d < p \end{cases}$$

```
void Image_Composite(const Mat &Img1, const Mat &Img2,int p){//E2-2
    Mat Tag;
    subtract(Img1,Img2,Tag);
    int r=Tag.rows,c=Tag.cols, inStep=Tag.step, Channels=Tag.channels(),Step=Tag.step;
    Mat Mask;Mask.create(r, c, CV_8UC3);
    int outStep=Mask.step,MaskChannels=Mask.channels();
    uchar* Tdata=Tag.data,* MaskD=Mask.data; int d;
    for(int i=0;i<r;i++){
        for(int j=0;j<c;j++){
            d=0;
            for(int k=0;k<Channels;++k){
                d+= Tdata[i * Step + j * Channels + k] * Tdata[i * Step + j * Channels + k];
            }
            d=sqrt(d);
            if(d>=p){
                for(int k=0;k<Channels;++k){
                    MaskD[i * outStep + j * Channels + k]=255;
                }
            }
            else{
                for(int k=0;k<Channels;++k){
                    MaskD[i * outStep + j * Channels + k]=0;
                }
            }
        }
    }
    imshow(WindowName,Mask);
}
```

原图：

效果：



此时发现必须自己不断调整阈值，来适应不同图片的最佳效果。

经过查询，找到一个由日本学者大津提出的图像灰度自适应的阈值分割算法(OTSU 算法)，这个方法通过不断遍历不同的阈值，计算不同阈值时对应的背景和前景之间的类内方差，当方差取得极大值时，对应的阈值就是所求的阈值。

该算法流程如下：

1. 计算 0~255 各灰阶对应的像素个数，保存至一个数组中，保存每个灰度值对应的像素数
2. 计算背景图像的平均灰度、背景图像像素数所占比例
3. 计算前景图像的平均灰度、前景图像像素数所占比例
4. 遍历 0~255 各灰阶，计算并寻找类间方差极大值

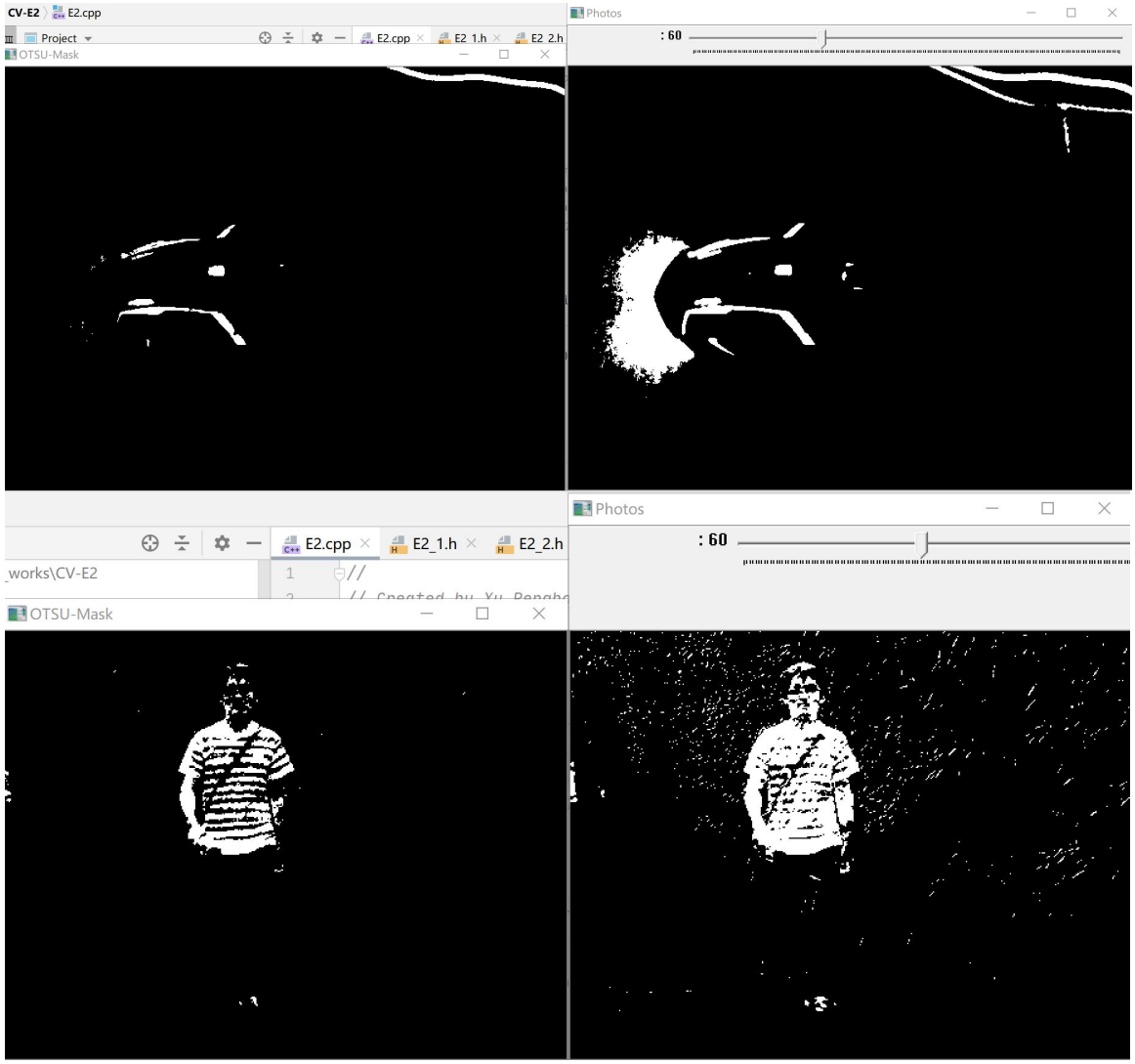
```
int OTSU(const Mat img)
{
    int T=0; //阈值
    double curValue=0; //类间方差中间值保存
    double w0=0; //前景像素点数所占比例
    double w1=0; //背景像素点数所占比例
    double u0=0; //前景平均灰度
```

```
double u1=0; //背景平均灰度
double Huidu[256]={0}; //灰度直方图，灰度值对应的像素点总数
uchar *data=img.data;
double totalNum= img.rows * img.cols; //像素总数
//计算灰度直方图
for(int i=0; i < img.rows; i++)
{
    for(int j=0; j < img.cols; j++)
    {
        Huidu[data[i * img.step + j]]++;
    }
}
for(int i=0;i<255;i++)
{
    w1=0;    u1=0;    w0=0;    u0=0;
    for(int j=0;j<=i;j++)
    {
        w1+=Huidu[j];
        u1+= j * Huidu[j];
    }
    if(w1==0) //背景部分像素点数为 0
    {
        continue;
    }
    u1=u1/w1; //背景平均灰度
    w1=w1/totalNum; // 背景所占比例
    for(int k=i+1;k<255;k++)
    {
        w0+=Huidu[k];   //前景部分总数
        u0+= k * Huidu[k]; //前景部分总灰度和
    }
    if(w0==0) //前景部分像素点数为 0
    {
        break;
    }
    u0=u0/w0; //前景平均灰度
    w0=w0/totalNum; // 前景部分所占比例

    double varValuel=w0*w1*(u1-u0)*(u1-u0); //当前类间方差计算
    if(curValue < varValuel)
    {
        curValue=varValuel;
        T=i;
    }
```

```
    }
    return T;
}
```

效果：



结果分析与体会：

只是通过简单的相减，设定阈值取出前景，容易受到影子或者树叶飘动等因素干扰出现噪声数据，需要自己调整阈值来观察效果。

大津法根据计算不同阈值时的类内方差然后找到取得极大值时的阈值。