题目要求：

结合计算机视觉课程知识，识别并计算出楼层数目、最高高度、最宽宽度、最长长度、窗户数目、窗户面积、整个建筑的体积（立方米数）。

实验设计：

边缘检测：

Sobel 算子：

Sobel 算子的优点是方法简单、处理速度快，并且所得的边缘光滑、连续。缺点是边缘较粗，由于处理时需对图像作二值化处理，故得到的边缘与阈值的选取也有很大的关系。

Laplacian 算子：

Laplacian 算子的优点是突现出图像中小的细节信息。缺点是对图像中的某些边缘产生双重响应。

Canny 算子：

Canny 算子的优点是能够尽可能多地标识出图像中的实际边缘，并且标识出的边缘要与实际图像中的实际边缘尽可能接近。缺点是图像中的边缘只能标识一次，并且可能错误地将图像噪声标识为边缘。（在此次实验中由于楼房的棱角和线条过多，导致 Canny 算子的处理效果并不理想，边缘检测步骤中被首先排除使用）

反色：

反色是与原色叠加可以变为白色的颜色，即用白色(RGB：255，255，255)减去原色的颜色。

```
Img=255-src
#反色
print(Img.shape)
cv2.imshow("Img",Img)
#输出反色图像
```

在此次实验中总的处理思路是框选出处理后图像中的白色部分（窗户）。由于部分处理效果的结果可能是黑色部分为窗户，所以需要通过反色处理，然后使用轮廓检测算法框选出正确的窗户。

平滑处理：

当像元放大后，图像的边界就会出现锯齿状，经过增加像元内插处理，加大像元分辨率，使图像细化，即平滑化处理。

```
img = cv.imread('laplacian1.jpg')
kernel = np.ones((20,20),np.float32)/50
dst = cv.filter2D(img,-1,kernel)
```

平滑处理效果：处理边缘检测后的图片，图片效果中会对边框进行加粗，以此突出每个边框。

形态学，开运算（先腐蚀后膨胀）去噪：

将开启和闭合结合起来可用来滤除噪声，首先对有噪声图象进行开启操作，可选择结构要素矩阵比噪声的尺寸大，因而开启的结果是将背景上的噪声去除。最后是对前一步得到的图像进行闭合操作，将图像上的噪声去掉。

轮廓检测函数：

轮廓检测指在包含目标和背景的数字图像中，忽略背景和目标内部的纹理以及噪声干扰的影响，采用一定的技术和方法来实现目标轮廓提取的过程。

```
contours, hierarchy = cv2.findContours(dst,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
#轮廓检测函数
#method:轮廓近似的方法
#cv2.CHAIN_APPROX_NONE:存储所有的轮廓点
#cv2.CHAIN_APPROX_SIMPLE:压缩水平，垂直和对角线段，只留下端点。 例如矩形轮廓可以用4个点编码。
#cv2.CHAIN_APPROX_TC89_L1,cv2.CHAIN_APPROX_TC89_KCOS:使用Teh-Chini chain近似算法
```

目前轮廓检测方法有两类，一类是利用传统的边缘检测算子检测目标轮廓，另一类是从人类视觉系统中提取可以使用的数学模型完成目标轮廓检测。

此次实验使用基于边缘检测的轮廓检测方法。这种方法是一种低层视觉行为，它主要定义了亮度、颜色等特征的低层突变，通过标识图像中亮度变化明显的点来完成边缘检测。边缘检测通常将图像与微分算子卷积。

提取图像 BGR 值：

在 RGB 颜色模式中，颜色由红色、绿色、蓝色混合而成。将颜色由一个十六进制符号来定义，这个符号由红色、绿色和蓝色的值组成（RGB）。

此次实验中特定区域与其余部分存在色差，比如楼顶的颜色浅于地面；窗户的颜色接近黑色，深于楼房墙壁。特定条件下可以选取特定的 BGR 范围进而选取特定区域，所以这个处理方式在实验中既用于窗户的选择，也用于楼房顶部的框选。

虽然这样的处理效果比较容易受照片拍摄和图像条件影响，但是在前光摄影条件下处理效果甚至优于所有常规的预处理。

轮廓检测——不规则图形——保留包围的最大面积

包围形状的面积计算：

属于轮廓检测函数类中的一类，在此次实验中使用该函数计算照片中框选区域的面积。

```
ares = cv2.contourArea(cont)
#计算包围形状的面积
```

部分噪点会影响框选效果，所以通过面积的大小来筛选一些不可能的方框，同时对每个框选区域进行计数。

```
if ares<750:   #(过滤面积小于75的形状)
    continue
count+=1
#总体计数加1
ares_avrg+=ares
```

在检测楼房顶部的区域时，由于存在部分背景的 BGR 值也在范围内，所以不可避免地会形成框选区域，所以判断所有闭合区域的面积，仅仅保留最大的面积，这样可以锁定楼顶区域。

```
theonlybuilding = image.copy()
contours = list(contours)
# print(type(contours))
contours.sort(key=len,reverse=True)
cv2.drawContours(theonlybuilding,[contours[0]],-1,(0,0,255),2)
```

大津算法：

图像分割中阈值选取的最佳算法，计算简单，不受图像亮度和对比度的影响，因此在数字图像处理上得到了广泛的应用。它是按图像的灰度特性，将图像分成背景和前景两部分。因方差是灰度分布均匀性的一种度量，背景和前景之间的类间方差越大，说明构成图像的两部分的差别越大，当部分前景错分为背景或部分背景错分为前景都会导致两部分差别变小。因此，使类间方差最大的分割意味着错分概率最小。

大津算法应用：

是求图像全局阈值的最佳方法，应用不言而喻，适用于大部分需要求图像全局阈值的场合。

优点：计算简单快速，不受图像亮度和对比度的影响。

缺点：对图像噪声敏感；只能针对单一目标分割；当目标和背景大小比例悬殊、类间方差函数可能呈现双峰或者多峰，这个时候效果不好。

局部大津算法与全局大津算法比较：

由于大津算法只能针对单一目标分割，所以利用局部阈值的大津算法进行图像二值化。这种处理下每个方框单独成为一个区域，减小目标和背景的大小比例。

```
#使用局部阈值的大津算法进行图像二值化
dst = cv2.adaptiveThreshold(gray,255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY,101, 1)

#全局大津算法，效果较差
#res ,dst = cv2.threshold(gray,0 ,255, cv2.THRESH_OTSU)
```

实验内容：

此次实验的目标是在第一问中建立出相关楼房的模型之后利用深度学习和计算机图形学的知识识别计算楼层的尺寸以及楼中多处细节的尺寸。

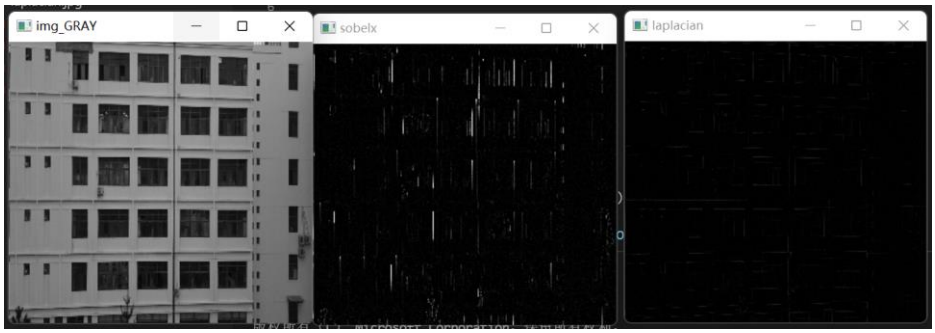
一、识别窗户与楼层：

思路：窗户和背景墙面的颜色深度不一样，同时照片中整体颜色偏浅，只有窗户接近深色。所以可以将预处理后的图片进行二值化，这样处理后窗户都为黑色，其余部分为白色。

如下图所示：

边缘检测：

边缘检测如下图所示：



可以看出由于楼房侧面线条比较多，简单的边缘检测效果并不理想。

算子的选择：

此次实验使用 sobel 和 laplacian 算子进行边缘检测。

sobel 算子：

该算子包含两组 3x3 的矩阵，分别为横向及纵向，将之与图像作平面卷积，即可分别得出横向及纵向的亮度差分近似值。如果以 A 代表原始图像，Gx 及 Gy 分别代表经横向及纵向边缘检测的图像，其公式如下：

$$\mathbf{G_x} = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G_y} = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

Sobel 算子的优点是方法简单、处理速度快，并且所得的边缘光滑、连续。

缺点是边缘较粗，由于处理时需作两值化处理，故得到的边缘与阈值的选取也有很大的关系。对于边缘模糊的部分，这种过程可以重复多次，也可得到较细的边缘(但不一定连续)。同时，这种方法还可保留低幅值的边缘。这样，既可提高定位精度，又可得到用其它方法难以得到的模糊边缘和微弱边缘。同时由于 sobel 算子对图片中灰度变化不够敏感，这样处理出来的结果会使图片的效果失真，同时不能突出窗户与楼体墙面的对比。

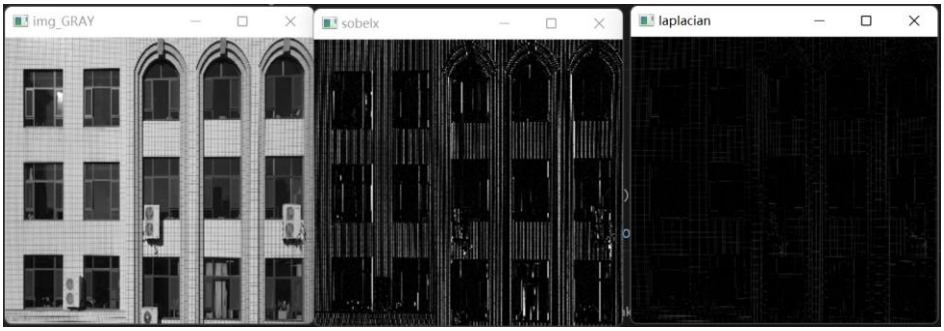
Laplacian 算子：

拉普拉斯算子是最简单的各向同性微分算子，它具有旋转不变性。一个二维图像函数的拉普拉斯变换是各向同性的二阶导数，定义为：

$$\mathrm{Laplace(f)} = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Laplacian 算子将在边缘处产生一个陡峭的零交叉， 能对任何走向的界线和线条进行锐化。这种简单的锐化方法既可以产生拉普拉斯锐化处理的效果，同时又能保留背景信息，将原始图像叠加到拉普拉斯变换的处理结果中去，可以使图像中的各灰度值得到保留，使灰度突变处的对比度得到增强，最终结果是在保留图像背景的前提下，突现出图像中小的细节信息。但其缺点是对图像中的某些边缘产生双重响应。

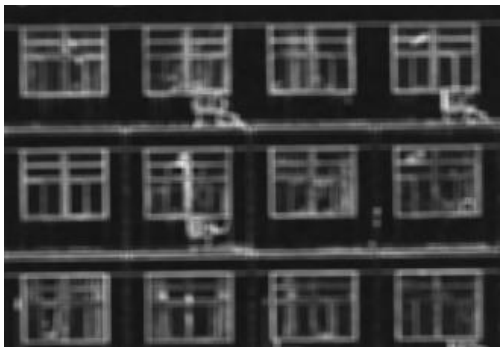
由于此次实验处理图片线条非常复杂，但是需要的数据仅仅是窗户这样存在棱角的特征物体，所以 laplacian 算子的处理效果相比起来最优。



在选择一张构图比较简单的图片之后，可以明显看出 sobel 算子将楼层中的砖块边缘也显示了出来，这不利于接下来的平滑处理和检测轮廓。相比之下 laplacian 算子在经过平滑处理之后效果如图所示：

```
img = cv.imread('laplacian1.jpg')
kernel = np.ones((20,20),np.float32)/50
dst = cv.filter2D(img,-1,kernel)
```

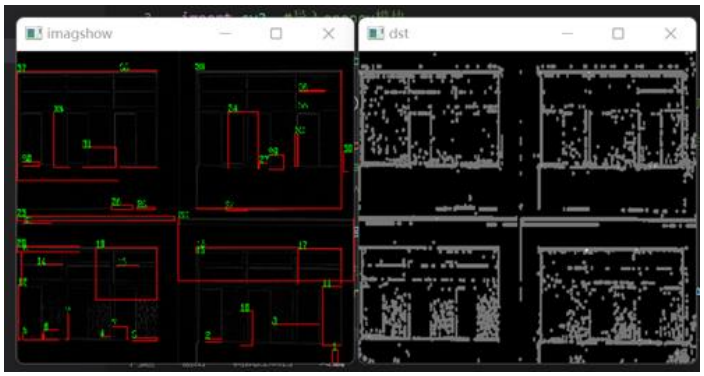
通过调整 kernel 中数组的行数和列数，会调整平滑处理的图像边缘宽度。处理‘test3.jpg’使用(20,20)效果最佳，效果如下图所示：



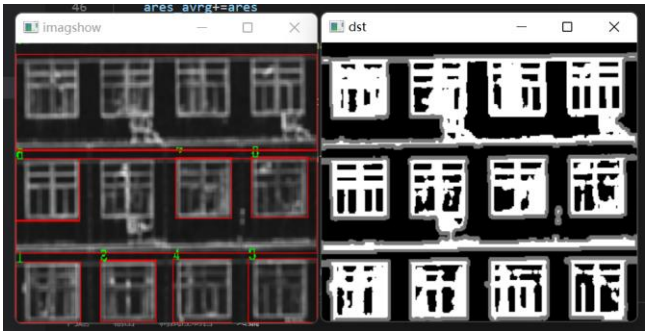
直接使用轮廓检测函数检测窗户效果如下图所示：

（进行图像二值化和去噪后使用轮廓检测函数）

（右侧图像边缘检测效果较差时会导致方框选取杂乱，调整边缘检测和筛选方框大小后再次尝试轮廓检测函数）

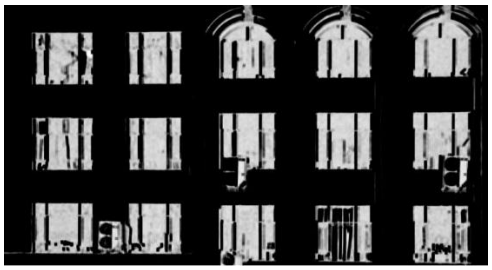


如下图所示，此次运行结果接近理想，但是由于楼层中存在空调机影响窗户选取导致选取不到楼层中部的一个窗户。



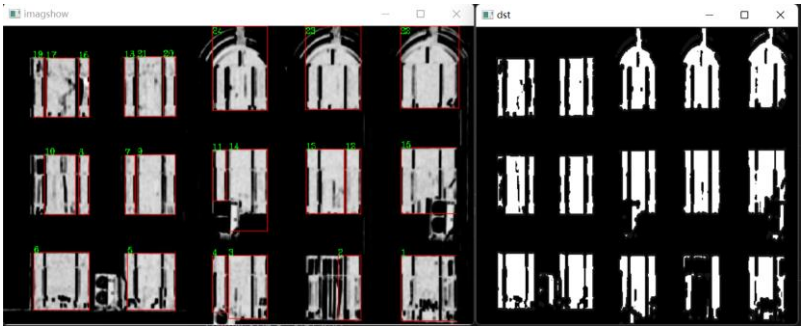
思路更新：

在昨晚楼房检测之后，我们小组尝试用识别楼房整体的思路来检测单个窗户，与前文中二值化加去噪处理相比起来，处理楼房的思路可以更加清晰地检测出窗户，这样的检测结果甚至可以将窗户中的铁条检测出来，这样会将整个窗户分成三个。测量步骤中同样具体测算每个窗户的长宽和面积。



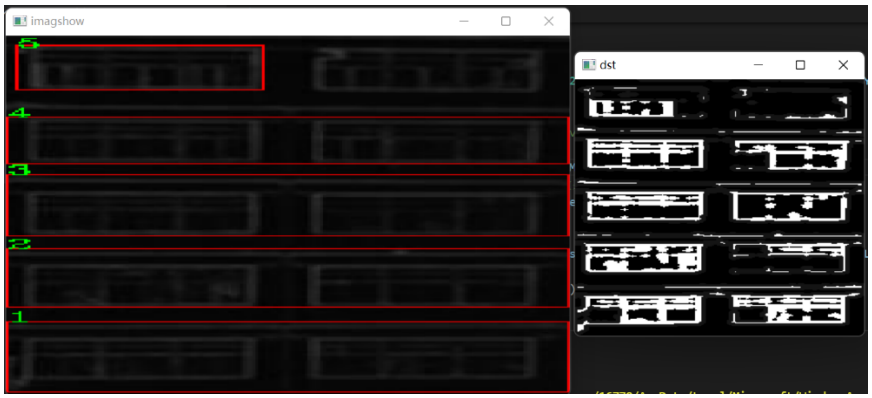
处理结果如下图所示：（图中共有 45 扇窗户，由于空调机和弧形窗户影响检测，总共可以检测出 35 扇）

检验结果：



识别楼层：

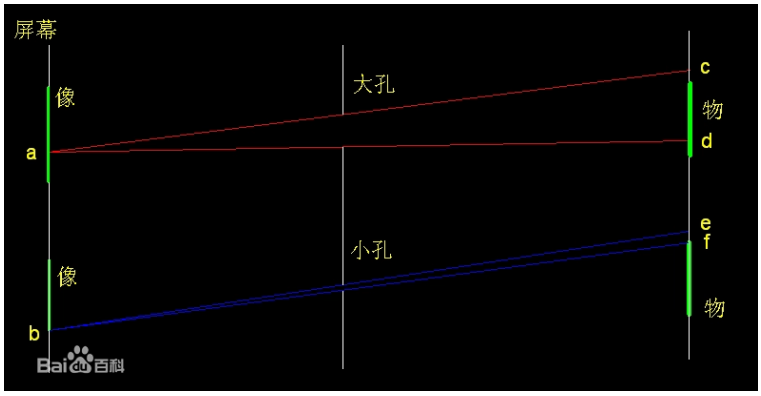
识别楼层的方法同样是使用轮廓检测函数，但是需要筛选出足够大的方框。通过标号可以清晰地看出楼层数。如下图所示:(图中五层楼都有可以检测出来，由于照片不够理想第五层不能全部检测出，但是这并不影响楼层识别与距离测量)



二、计算面积：

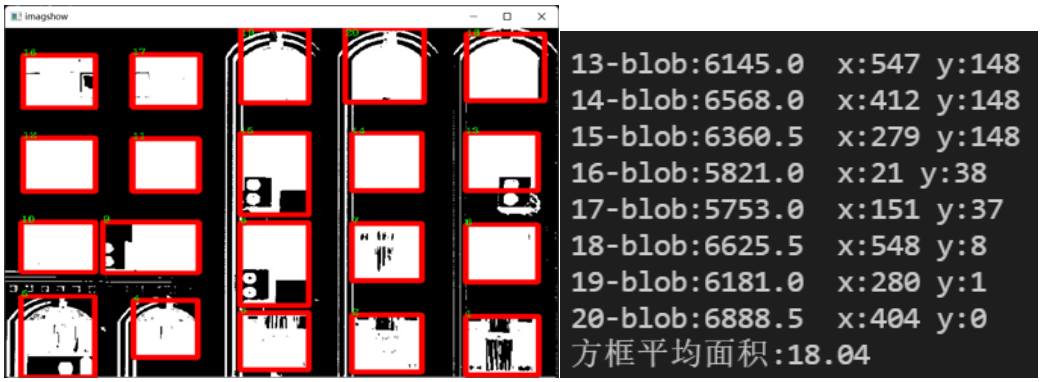
小孔成像：

拍照的过程在数学上相当于投影，损失了一个维度参数，因此没有一个简单普适的算法可以直接得到原始参数。必须有一个已知参数值（比如说摄像头对地面高度、两次拍摄点距离、固定参照物距离等等）。所以为了达成接近的结果，可以把镜头系统简化为小孔成像，这样计算起来大大简化。

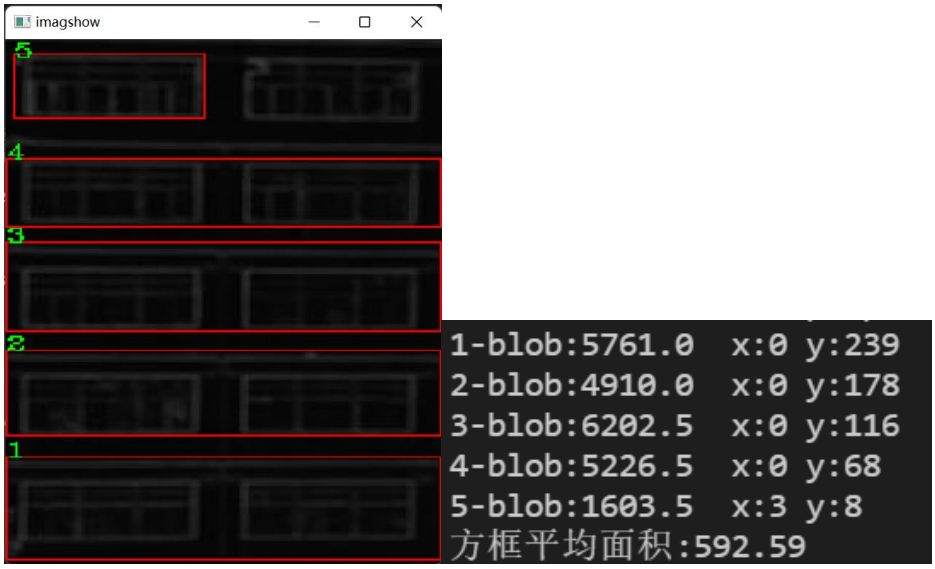


用一个带有小孔的板遮挡在墙体与物之间，墙体上就会形成物的倒立的实像，我们把这样的一种现象叫小孔成像。前后移动中间的板，墙体上像的大小也会随之发生变化。通过像素点算出图像中方框大小后带入相机拍摄地离建筑物的距离后可以推算出楼层与窗户的实际大小。具体过程是利用小孔成像原理：拍照的时候焦距代表底片到镜头的距离和镜头到楼房的距离，在处理过程中已经得知出现方框的长度和面积，这样可以算出窗户的面积。

窗户面积结果如下图所示：



楼层尺寸结果如下图所示：



通过小孔成像原理得出的窗口面积是：

商学院正常窗户：

长度	宽度	面积
6.0m（3 扇并排）	1.9m	12m²（处理图像存在误差）

学生宿舍 5 号楼与：

长度	宽度	面积
3.2m	1.9m	6m²（处理图像存在误差）

网络楼正常窗户

长度	宽度	面积
2.5m	1.9m	5m²（处理图像存在误差）

三、识别楼房整体

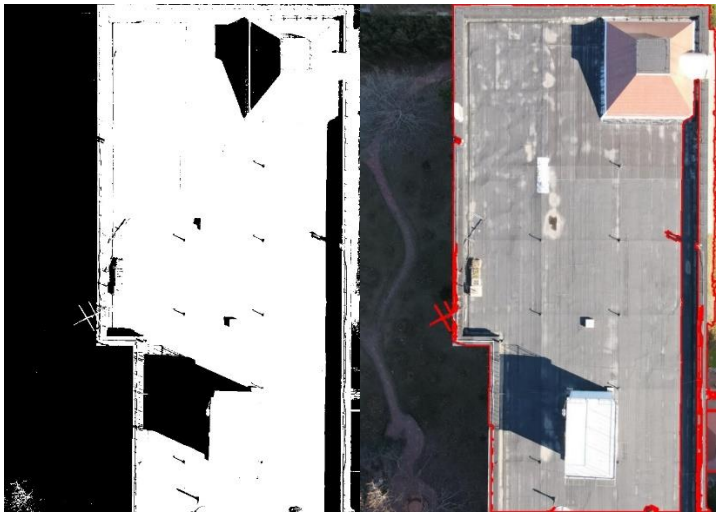
思路：楼房整体的识别与窗户的识别不同，由于楼体的规整程度远远小于窗户，并且地面和楼房的色差较小，所以轮廓检测函数不能很好的将整栋楼标记出来。

因此，检测楼栋使用全新的检测思路：先通过课上实验做出的程序检测出楼

顶某个点的 BGR 值，然后设定阈值范围后可以选择出接近楼房颜色的所有位置。最后选择框出的若干方框中面积最大的一个。

运行结果如下图所示：

检测网络楼：



在检测商学院顶层时会遇到周围色差较小导致会将部分马路囊括进楼房顶层的情况，调整 BGR 的范围可以避免这种情况。

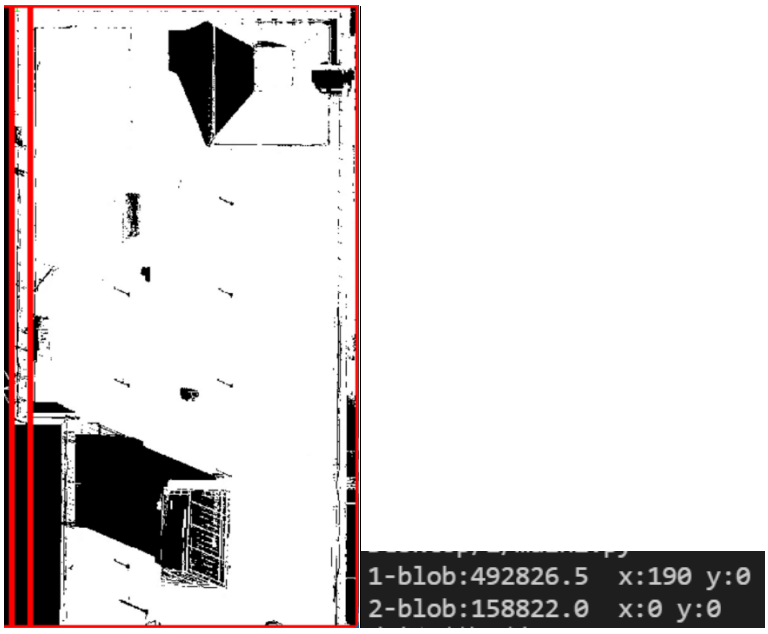
调整阈值之后：

检测学生宿舍 5 号楼：

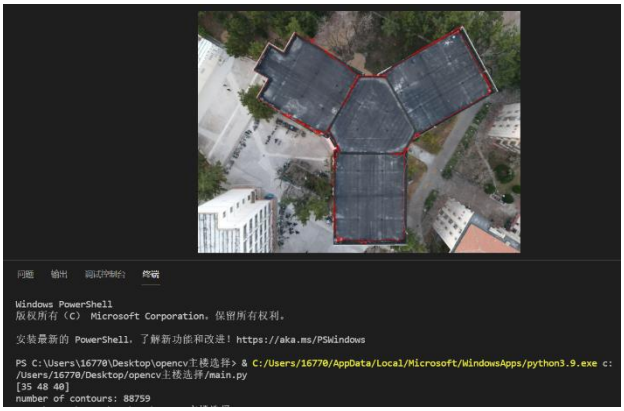
四、计算楼房整体长度最值，楼房的面积与体积

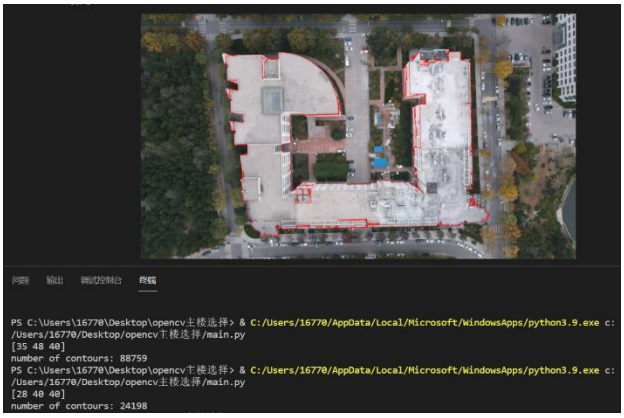
思路：

对于形状规则的楼房处理比较简单，使用轮廓检测函数处理二值化图像即可。对于实验中存在部分形状不规则的楼房，通过轮廓检测函数将整栋楼框选出来得出单个数据最值，然后将多个方框的值进行比较筛选才能得出楼房总体最值。



楼房整体的体积算法是通过楼体检测得出每个楼房的底面积（小孔成像原理）乘以楼房的高（由窗户检测得出）最终得到楼房体积。





商学院：

底面积	高度	体积
7600 m²（取整）	31m	23.6 万 m³

网络楼：

底面积	高度	体积
700m²（取整）	21m	1.47 万 m³

学生宿舍 5 号楼：

底面积	高度	体积
1800m²（取整）	22m	3.96 万 m³

实验结果：

楼层数目：

	商学院	学生宿舍 5 号楼	网络楼
实验结果	5 层	6 层	6 层
实际结果	5 层	6 层	6 层

窗户数目：

	商学院	学生宿舍 5 号楼	网络楼
实验结果	60 扇	87 扇	44 扇
实际结果	60 扇	100 扇	44 扇

窗户尺寸：

	商学院	学生宿舍 5 号楼	网络楼
实验结果	长 6.0m—宽 1.9m	长 3.2m—宽 1.9m	长 2.5m—宽 1.9m
实际结果	长 4.7m—宽 2.1m	长 3.4m—宽 1.7m	长 2.6m—宽 2.3m

窗户面积：

	商学院	学生宿舍 5 号楼	网络楼
实验结果	12 m^2	6m	5m
实际结果	9.9m^2	5.9m	5.9m

最长长度：

	商学院	学生宿舍 5 号楼	网络楼
实验结果	120m	35m	40m
实际结果	126m	36.7m	45.7m

最宽宽度：

	商学院	学生宿舍 5 号楼	网络楼
实验结果	60m	15m	18m
实际结果	59.7m	16.3m	17.9m

最高高度：

	商学院	学生宿舍 5 号楼	网络楼
实验结果	31m	22m	19m
实际结果	26.2m	21.1m	24m

建筑物底面积：

	商学院	学生宿舍 5 号楼	网络楼
实验结果	7600 m^2	1800 m^2	700 m^2
实际结果	7630 m^2	1599 m^2	743 m^2

整个建筑的体积：

	商学院	学生宿舍 5 号楼	网络楼
实验结果	23.6 万 m^3	3.96 万 m^3	1.47 万 m^3
实际结果	19.9 万 m^3	3.37 万 m^3	1.78 万 m^3