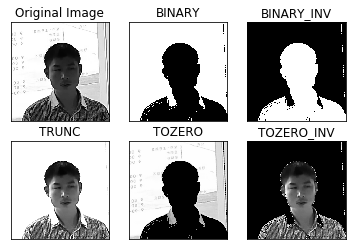
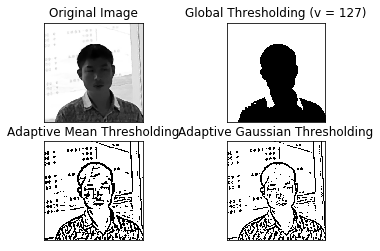
# 图像阈值

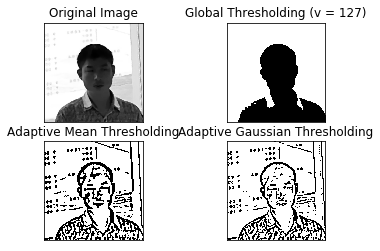
简单阈值



自适应阈值

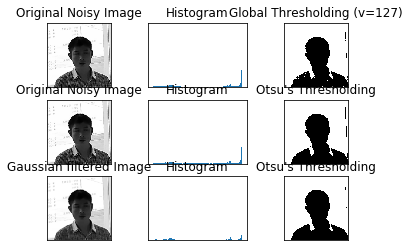


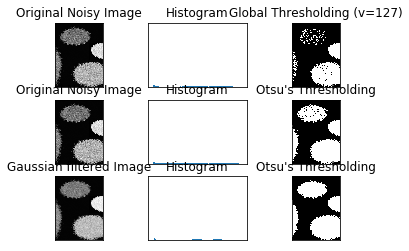
• Block Size - 邻域大小（用来计算阈值的区域大小）=11



• Block Size - 邻域大小（用来计算阈值的区域大小）=15

Otsu’ ’s 二值化





**Otsu’ ’s 二值化是如何工作的？**  
　　在这一部分我们会演示怎样使用 Python 来实现 Otsu 二值化算法，从而告诉大家它是如何工作的。如果你不感兴趣的话可以跳过这一节。因为是双峰图，Otsu 算法就是要找到一个阈值（t）, 使得同一类加权方差最小，需要满足下列关系式：

\sigma_w^2(t) = q_1(t)\sigma_1^2(t)+q_2(t)\sigma_2^2(t)  
　　其中：

q_1(t) = \sum_{i=1}^{t} P(i) \quad \& \quad q_1(t) = \sum_{i=t+1}^{I} P(i)

\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_1(t)} \quad \& \quad \mu_2(t) = \sum_{i=t+1}^{I} \frac{iP(i)}{q_2(t)}

\sigma_1^2(t) = \sum_{i=1}^{t} [i-\mu_1(t)]^2 \frac{P(i)}{q_1(t)} \quad \& \quad \sigma_2^2(t) = \sum_{i=t+1}^{I} [i-\mu_1(t)]^2 \frac{P(i)}{q_2(t)}

代码实现部分

img = cv2.imread('noisy2.png',0)

blur = cv2.GaussianBlur(img,(5,5),0)

# find normalized\_histogram, and its cumulative distribution function

hist = cv2.calcHist([blur],[0],None,[256],[0,256])

hist\_norm = hist.ravel()/hist.max()

Q = hist\_norm.cumsum()

bins = np.arange(256)

fn\_min = np.inf

thresh = -1

for i in xrange(1,256):

p1,p2 = np.hsplit(hist\_norm,[i]) # probabilities

q1,q2 = Q[i],Q[255]-Q[i] # cum sum of classes

b1,b2 = np.hsplit(bins,[i]) # weights

# finding means and variances

m1,m2 = np.sum(p1\*b1)/q1, np.sum(p2\*b2)/q2

v1,v2 = np.sum(((b1-m1)\*\*2)\*p1)/q1,np.sum(((b2-m2)\*\*2)\*p2)/q2

# calculates the minimization function

fn = v1\*q1 + v2\*q2

if fn < fn\_min:

fn\_min = fn

thresh = i

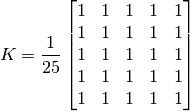
# find otsu's threshold value with OpenCV function

ret, otsu = cv2.threshold(blur,0,255,cv2.THRESH\_BINARY+cv2.THRESH\_OTSU)

prin(thresh,ret)

# 图像平滑

2D卷积

**2D 卷积**  
　　与一维信号一样，我们也可以对 2D 图像实施低通滤波（LPF），高通滤波（HPF）等。LPF 帮助我们去除噪音，模糊图像。HPF 帮助我们找到图像的边缘  
OpenCV 提供的函数 cv.filter2D() 可以让我们对一幅图像进行卷积操作。下面我们将对一幅图像使用平均滤波器。下面是一个 5x5 的平均滤波器核：  
　　　　　　  
操作如下：将核放在图像的一个像素 A 上，求与核对应的图像上 25（5x5）个像素的和，在取平均数，用这个平均数替代像素 A 的值。重复以上操作直到将图像的每一个像素值都更新一边。代码如下，运行一下吧。





# 19 Canny 边缘检测

**目标**  
　　• 了解 Canny 边缘检测的概念  
　　• 学习函数 cv2.Canny()

**19.1 原理**  
　　Canny 边缘检测是一种非常流行的边缘检测算法，是 John F.Canny 在1986 年提出的。它是一个有很多步构成的算法，我们接下来会逐步介绍。

**19.1.1 噪声去除**  
　　由于边缘检测很容易受到噪声影响，所以第一步是使用 5x5 的高斯滤波器去除噪声，这个前面我们已经学过了。

**19.1.2 计算图像梯度**  
　　对平滑后的图像使用 Sobel 算子计算水平方向和竖直方向的一阶导数（图像梯度）（Gx 和 Gy）。根据得到的这两幅梯度图（Gx 和 Gy）找到边界的梯度和方向，公式如下：  
　　　　　　Edge\_Gradient \; (G) = \sqrt{G_x^2 + G_y^2}

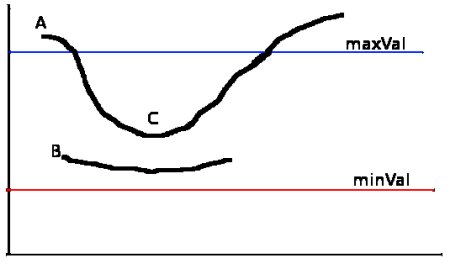
Angle \; (\theta) = \tan^{-1} \bigg(\frac{G_y}{G_x}\bigg)  
梯度的方向一般总是与边界垂直。梯度方向被归为四类：垂直，水平，和两个对角线。

**19.1.3 非极大值抑制**  
　　在获得梯度的方向和大小之后，应该对整幅图像做一个扫描，去除那些非边界上的点。对每一个像素进行检查，看这个点的梯度是不是周围具有相同梯度方向的点中最大的。如下图所示：



现在你得到的是一个包含“窄边界”的二值图像。

**19.1.4 滞后阈值**  
　　现在要确定那些边界才是真正的边界。这时我们需要设置两个阈值：minVal 和 maxVal。当图像的灰度梯度高于 maxVal 时被认为是真的边界，那些低于 minVal 的边界会被抛弃。如果介于两者之间的话，就要看这个点是否与某个被确定为真正的边界点相连，如果是就认为它也是边界点，如果不是就抛弃。如下图：

  
A 高于阈值 maxVal 所以是真正的边界点，C 虽然低于 maxVal 但高于minVal 并且与 A 相连，所以也被认为是真正的边界点。而 B 就会被抛弃，因为他不仅低于 maxVal 而且不与真正的边界点相连。所以选择合适的 maxVal和 minVal 对于能否得到好的结果非常重要。  
在这一步一些小的噪声点也会被除去，因为我们假设边界都是一些长的线段。