1、获取每个数字的区域

1.1、图像阈值处理

为了进一步标准化处理数字区域，我们先把图像按比例缩放到高度为100像素。

height, width = output.shape[:2]

print(height,width)

output = cv2.resize(output, (int(100\*width/height), 100))

plt.imshow(output,cmap='gray')

print(output.shape)

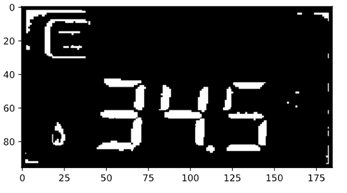
然后，我们先通过阈值法分割图像。我们将数字部分作为前景，因此使用反二值化的阈值处理，将数字部分作为白色前景。根据图像的灰度值，选取45作为阈值。

# 用阈值法分割图像，然后执行形态学开运算清理分割的图像

ret, thresh = cv2.threshold(output, 62, 255,    cv2.THRESH\_BINARY\_INV )

plt.imshow(thresh, cmap='gray')

可以看到获得的数字区域的二值化图像：



观察到数字本身的各个笔划之间有间隔，那么我们需要通过形态学的操作清理图像，通过闭操作把数字笔划之间连接到一起。这里我们通过3\*5的核，完成图像的闭操作。这里核的大小可以通过不断尝试获取。要求数字的笔划相连，但是和其他部分不相连。

kernel = cv2.getStructuringElement(cv2.MORPH\_RECT, (3, 5))

closing = cv2.morphologyEx(thresh, cv2.MORPH\_CLOSE, kernel)

plt.figure(figsize=(10,10))

plt.imshow(closing, cmap='gray')

可以看到效果



1.2、提取数字部分

然后，进一步提取轮廓，并对每一个轮廓获取其边界矩形，如果矩形的宽在20到35像素之间，高在30到50之间，则认为是要找的数字区域。

# 轮廓提取

contours, hierarchy = cv2.findContours(closing.copy(), cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

#数据区域矩阵的列表

digitCnts = []

for c in range(len(contours)):

    #获取轮廓的边界矩形

    x, y, w, h = cv2.boundingRect(contours[c])

    # 如果轮廓大小合适，就是要找的数字区域

    if (w >= 20 and w <=35) and (h >= 30 and h <= 50):

        digitCnts.append([x,y,w,h])

        cv2.rectangle(thresh, (x, y), (x + w, y + h), (255, 255, 255), 1)

print(digitCnts)

plt.imshow(thresh,cmap="gray")

获得列表

[[85, 48, 31, 39], [124, 46, 28, 44], [46, 44, 31, 46]]

可以看出来确实是选取了数字区域

