

计算机视觉与模式识别

2021 年 03 月 25 日

一、实验内容

GrabCut 原始文献（公式 11）和所给代码中使用了欧氏距离作为像素间相似度的评价指标，修改公式，用其他相似度度量方式进行分割；计算图像的局部特征，再用这些特征计算相似度。在不少于 3 张的个人图像上对比修改前后的算法性能。

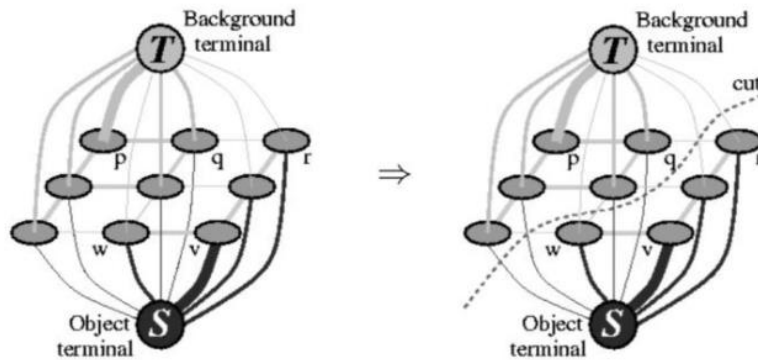
$$V(\underline{\alpha}, \mathbf{z}) = \gamma \sum_{(m,n) \in \mathbf{C}} [\alpha_n \neq \alpha_m] \exp -\beta \|z_m - z_n\|^2. \quad (11)$$

二、实验原理

2.1 算法原理

GrabCut 算法由 GraphCut 算法改进而来。首先得简要说明前者的原理，再看后者的原理。

GraphCut 是一种让用户交互，指定图像中部分前景和背景区域，进行自动分割前后景的算法。用户指定了少量的前景和背景后，用像素建立图，每个节点是一个像素，除此之外，还有两个特别的节点 T 和 S 节点，分别代表背景和前景；然后在所有节点之间建立连接，像素之间的连接叫做 n-links，像素到 T 或 S 节点的连接是 t-links；为这些连接赋予权重（能量），t-links 的能量表示一个像素与 T 或 S 的相似度，已经被标注为前景或者背景的像素对应的 t-links 很粗；n-links 的能量约束着像素之间是否会被划分为不同的区域，n-links 的能量（权重）越小，表明像素之间越不相似，越可能被分割开；赋予权重后，采用最小割（mini-cut）算法，将前后景分开。



图表 1: GraphCut 原理简图

GraphCut 算法采用的是用户标注的前景和背景的灰度统计直方图作为先验信息，这样就不适合用于彩色图片了。

GrabCut 做了不少改进。它不采用灰度统计直方图，而是采用高斯混合模型（GMM）来作为统计先验知识，能用于彩色图；用户标注更简单；不同于 GraphCut 的单词 min-cut，GrabCut 采用了迭代进行 min-cut，每次最小割都会标注出新的前景和背景，用该结果当作下一次迭代的基础，直到收敛；用户可以对不完美的地方进行修补。

2.2 改变的代码

2.2.1 余弦相似度

原本是计算像素之间的欧氏距离作为度量，现在通过计算像素之间的余弦相似度来衡量，由于余弦相似度是越大越相似，而欧氏距离是越大越不相似，刚好是反过来的，所以采用 1-cosine-similarity 的方法：

```
def calc_beta_smoothness(self):
    # calculate cosine similarity of different pixels
    t11, t12 = self.img[:, 1:], self.img[:, :-1]
    t21, t22 = self.img[1:, 1:], self.img[:-1, :-1]
    t31, t32 = self.img[1:, :], self.img[:-1, :]
    t41, t42 = self.img[:, :-1], self.img[:, 1:]

    # Matrix: length of pixels
    length1 = (np.linalg.norm(t11, axis=2) * np.linalg.norm(t12, axis=2))
    length2 = (np.linalg.norm(t21, axis=2) * np.linalg.norm(t22, axis=2))
    length3 = (np.linalg.norm(t31, axis=2) * np.linalg.norm(t32, axis=2))
    length4 = (np.linalg.norm(t41, axis=2) * np.linalg.norm(t42, axis=2))

    # make sure their values are not zero, because a dividend must not be 0
    length1 = np.where(length1==0, 1, length1)
    length2 = np.where(length2==0, 1, length2)
    length3 = np.where(length3==0, 1, length3)
    length4 = np.where(length4==0, 1, length4)

    ''' get 1 - cosine-similarity, because original formula is Euclidean distance
    (which means the bigger the pixel distance is, the less similar they are.
    Cosine-similarity is to the contrary, so 1 - cos would be better)'''
    _left_diff = 1 - ((np.sum(t11 * t12, axis=2))) / length1
    _upleft_diff = 1 - ((np.sum(t21 * t22, axis=2))) / length2
    _up_diff = 1 - ((np.sum(t31 * t32, axis=2))) / length3
    _upright_diff = 1 - ((np.sum(t41 * t42, axis=2))) / length4
```

```

self.beta = np.sum((_left_diff)) + np.sum((_upleft_diff)) + np.sum((_up_diff)) + np.sum((_upright_diff))
self.beta = 1 / (2 * self.beta / (
    # Each pixel has 4 neighbors (left, upleft, up, upright)
    4 * self.cols * self.rows
    # The 1st column doesn't have left, upleft and the last column doesn't have upright
    - 3 * self.cols
    - 3 * self.rows # The first row doesn't have upleft, up and upright
    + 2)) # The first and last pixels in the 1st row are removed twice
print('Beta:', self.beta)

# Smoothness term V described in formula (11)
self.left_v = self.gamma * np.exp(-self.beta * _left_diff)
self.upleft_v = self.gamma / np.sqrt(2) * np.exp(-self.beta * _upleft_diff)
self.up_v = self.gamma * np.exp(-self.beta * _up_diff)
self.upright_v = self.gamma / np.sqrt(2) * np.exp(-self.beta * _upright_diff)

```

图表 2：余弦相似度修改代码

2.2.2 L1 范数

这个就比较简单了，只需要在改为像素相减，绝对值相加即可，虽然改动简单，但是效果也不错。

```

def calc_beta_smoothness(self):
    # calculate L1-norm of different pixels
    t11, t12 = self.img[:, 1:], self.img[:, :-1]
    t21, t22 = self.img[1:, 1:], self.img[:-1, :-1]
    t31, t32 = self.img[1:, :], self.img[:-1, :]
    t41, t42 = self.img[1:, :-1], self.img[:-1, 1:]

    # get L1-NORM of pixels
    _left_diff = np.sum(np.abs(t11 - t12), axis=2)
    _upleft_diff = np.sum(np.abs(t21 - t22), axis=2)
    _up_diff = np.sum(np.abs(t31 - t32), axis=2)
    _upright_diff = np.sum(np.abs(t41 - t42), axis=2)

    self.beta = np.sum((_left_diff)) + np.sum((_upleft_diff)) + np.sum((_up_diff)) + np.sum((_upright_diff))
    self.beta = 1 / (2 * self.beta / (
        # Each pixel has 4 neighbors (left, upleft, up, upright)
        4 * self.cols * self.rows
        # The 1st column doesn't have left, upleft and the last column doesn't have upright
        - 3 * self.cols
        - 3 * self.rows # The first row doesn't have upleft, up and upright
        + 2)) # The first and last pixels in the 1st row are removed twice
    print('Beta:', self.beta)

    # Smoothness term V described in formula (11)
    self.left_v = self.gamma * np.exp(-self.beta * _left_diff)
    self.upleft_v = self.gamma / np.sqrt(2) * np.exp(-self.beta * _upleft_diff)
    self.up_v = self.gamma * np.exp(-self.beta * _up_diff)
    self.upright_v = self.gamma / np.sqrt(2) * np.exp(-self.beta * _upright_diff)

```

图表 3：L1 范数修改代码

2.2.3 切比雪夫距离

这个也比较简单，是像素之间相减，然后取绝对值最大的作为衡量。除去下图部分对应的代码不一样外，其余和 L1 的代码一样。

```
# get L1-NORM of pixels
_left_diff      = np.max(np.abs(t11 - t12), axis=2)
_upleft_diff    = np.max(np.abs(t21 - t22), axis=2)
_up_diff        = np.max(np.abs(t31 - t32), axis=2)
_upright_diff   = np.max(np.abs(t41 - t42), axis=2)
```

图表 4：切比雪夫距离代码

2.2.4 LBP 特征提取

这里考虑的是不直接用原图计算像素间的相似度，而是先计算 LBP 特征，然后用 LBP 图像去计算相似度。

```
# calculate lbp features in 3 dimensions
LBP_features = np.zeros((self.img.shape), np.float) # must be float, not int, or an error would occur
LBP_features[:, :, 0] = local_binary_pattern(self.img[:, :, 0], 8, 1)
LBP_features[:, :, 1] = local_binary_pattern(self.img[:, :, 1], 8, 1)
LBP_features[:, :, 2] = local_binary_pattern(self.img[:, :, 2], 8, 1)

cv.imshow('lbp', LBP_features / 255.0)

_left_diff = LBP_features[:, 1:] - LBP_features[:, :-1]
_upleft_diff = LBP_features[1:, 1:] - LBP_features[:-1, :-1]
_up_diff = LBP_features[1:, :] - LBP_features[:-1, :]
_upright_diff = LBP_features[1:, :-1] - LBP_features[:-1, 1:]
```

图表 5：LBP 特征提取代码

2.2.5 HOG 特征提取

这里考虑不直接用原图计算像素间的相似度，而是先计算 HOG 特征，然后用 HOG 图像去计算相似度。

```
# calculate HOG features in 3 dimensions
def HOG(img):
    normalized_blocks, hog_image = hog(img,
        orientations=9,
        pixels_per_cell=(8,8),
        cells_per_block=(2,2),
        block_norm='L2',
        feature_vector=True,
        visualize=True,
        multichannel=False
    )
    return normalized_blocks, hog_image
HOG_features = np.zeros((self.img.shape), np.float) # must be float, not int, or an error would occur
_, HOG_features[:, :, 0] = HOG(self.img[:, :, 0])
_, HOG_features[:, :, 1] = HOG(self.img[:, :, 1])
_, HOG_features[:, :, 2] = HOG(self.img[:, :, 2])

cv.imshow('HOG', HOG_features)

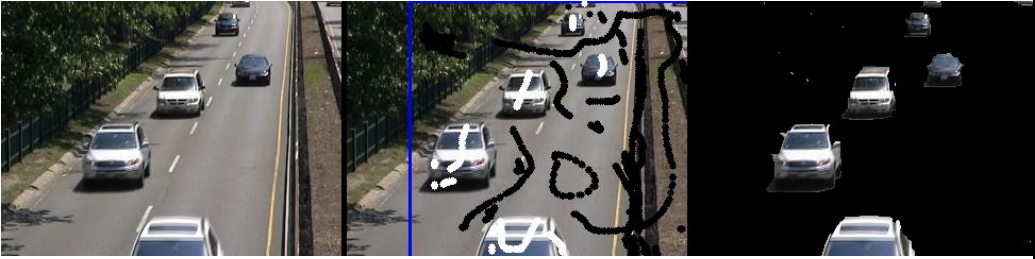
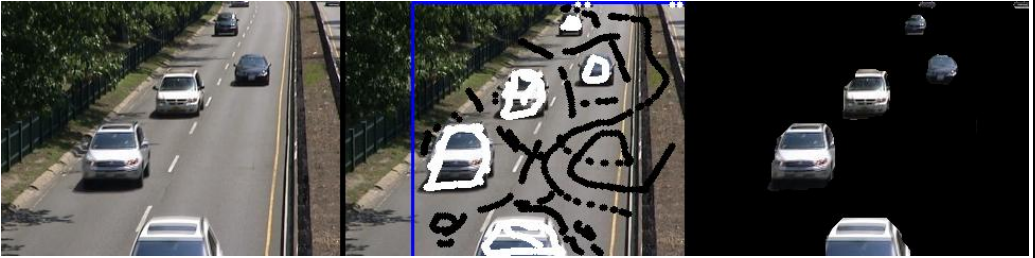

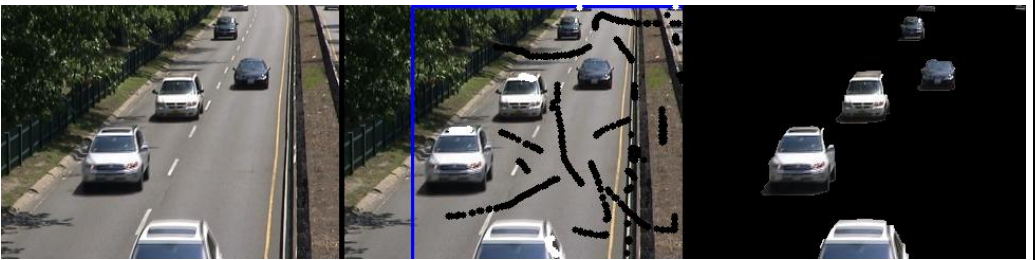
_left_diff = HOG_features[:, 1:] - HOG_features[:, :-1]
_upleft_diff = HOG_features[1:, 1:] - HOG_features[:-1, :-1]
_up_diff = HOG_features[1:, :] - HOG_features[:-1, :]
_upright_diff = HOG_features[1:, :-1] - HOG_features[:-1, 1:]
```

图表 6：HOG 特征提取代码

三、实验结果与分析

3.1 更换相似度计算公式结果对比


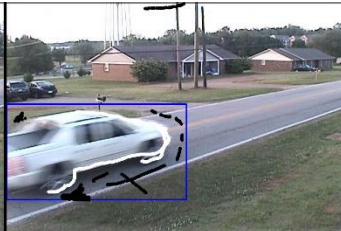







3.1.1 实验结果截图

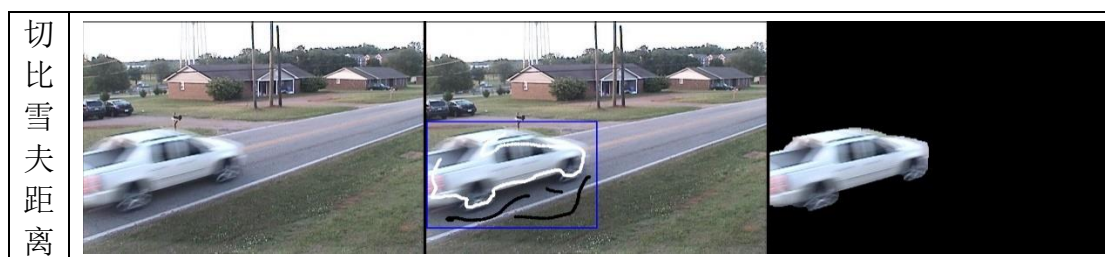
欧氏距离	
余弦相似度	
L1范数	
切比雪夫距离	

表格 1：图片一

欧氏距离			
余弦相似度			
L1范数			
切比雪夫距离			

表格 2: 图片二

欧氏距离			
余弦相似度			
L1范数			



表格 3：图片三

3.1.2 实验结果数据

欧氏距离	<pre> ----- Confusion matrix ----- Actual\Predicted Foreground Background Total Foreground 9505 2157 11662 Background 91 65047 65138 Total 9596 67204 76800 ----- Mean Pixel Accuracy(MPA): 90.68 % Accuracy: 97.07 % Precision: 99.05 % Recall: 81.5 % F1-socre: 89.42 % </pre>
余弦相似度	<pre> ----- Confusion matrix ----- Actual\Predicted Foreground Background Total Foreground 8715 2947 11662 Background 36 65102 65138 Total 8751 68049 76800 ----- Mean Pixel Accuracy(MPA): 87.34 % Accuracy: 96.12 % Precision: 99.59 % Recall: 74.73 % F1-socre: 85.39 % </pre>
L1范数	<pre> ----- Confusion matrix ----- Actual\Predicted Foreground Background Total Foreground 9271 2391 11662 Background 44 65094 65138 Total 9315 67485 76800 ----- Mean Pixel Accuracy(MPA): 89.71 % Accuracy: 96.83 % Precision: 99.53 % Recall: 79.5 % F1-socre: 88.39 % </pre>

切比雪夫距离	----- Confusion matrix -----			
	Actual\Predicted	Foreground	Background	Total
	Foreground	9411	2251	11662
	Background	38	65100	65138
	Total	9449	67351	76800

	Mean Pixel Accuracy(MPA):	90.32 %		
	Accuracy:	97.02 %		
	Precision:	99.6 %		
	Recall:	80.7 %		
	F1-score:	89.16 %		

表格 4：图片一

欧氏距离	----- Confusion matrix -----			
	Actual\Predicted	Foreground	Background	Total
	Foreground	8520	748	9268
	Background	278	76854	77132
	Total	8798	77602	86400

	Mean Pixel Accuracy(MPA):	95.78 %		
	Accuracy:	98.81 %		
	Precision:	96.84 %		
	Recall:	91.93 %		
	F1-score:	94.32 %		

余弦相似度	----- Confusion matrix -----			
	Actual\Predicted	Foreground	Background	Total
	Foreground	8307	961	9268
	Background	200	76932	77132
	Total	8507	77893	86400

	Mean Pixel Accuracy(MPA):	94.69 %		
	Accuracy:	98.66 %		
	Precision:	97.65 %		
	Recall:	89.63 %		
	F1-score:	93.47 %		

L 1 范 数	<pre> ----- Confusion matrix ----- Actual\Predicted Foreground Background Total Foreground 8532 736 9268 Background 128 77004 77132 Total 8660 77740 86400 ----- Mean Pixel Accuracy(MPA): 95.95 % Accuracy: 99.0 % Precision: 98.52 % Recall: 92.06 % F1-socre: 95.18 % </pre>
	切 比 雪 夫 距 离

表格 5: 图片二

欧 氏 距 离	<pre> ----- Confusion matrix ----- Actual\Predicted Foreground Background Total Foreground 39616 6804 46420 Background 18 291482 291500 Total 39634 298286 337920 ----- Mean Pixel Accuracy(MPA): 92.67 % Accuracy: 97.98 % Precision: 99.95 % Recall: 85.34 % F1-socre: 92.07 % </pre>

余弦相似度	<pre> ----- Confusion matrix ----- Actual\Predicted Foreground Background Total Foreground 41291 5129 46420 Background 270 291230 291500 Total 41561 296359 337920 ----- Mean Pixel Accuracy(MPA): 94.43 % Accuracy: 98.4 % Precision: 99.35 % Recall: 88.95 % F1-score: 93.86 % </pre>
L1范数	<pre> ----- Confusion matrix ----- Actual\Predicted Foreground Background Total Foreground 38180 8240 46420 Background 18 291482 291500 Total 38198 299722 337920 ----- Mean Pixel Accuracy(MPA): 91.12 % Accuracy: 97.56 % Precision: 99.95 % Recall: 82.25 % F1-score: 90.24 % </pre>
切比雪夫距离	<pre> ----- Confusion matrix ----- Actual\Predicted Foreground Background Total Foreground 38195 8225 46420 Background 18 291482 291500 Total 38213 299707 337920 ----- Mean Pixel Accuracy(MPA): 91.14 % Accuracy: 97.56 % Precision: 99.95 % Recall: 82.28 % F1-score: 90.26 % </pre>

表格 6：图片三

3.1.3 实验结果分析

从视觉上来看，三者的差异并不大，差别主要体现在边界上是否留有间隙、以及用户标注量的差别。第一张多辆汽车的图中，场景复杂，欧氏距离以及切比雪夫距离需要的标注量稍微比其他的小，余弦相似度的标注量最多，最后效果也没有那么好，从最后的评估数据来看，无论是准确率、召回率等，余弦相似度均表现最差；最好的是切比雪夫相似度衡量。

而在第二张图中，切比雪夫和欧氏距离的视觉表现则没有那么好，主要是有

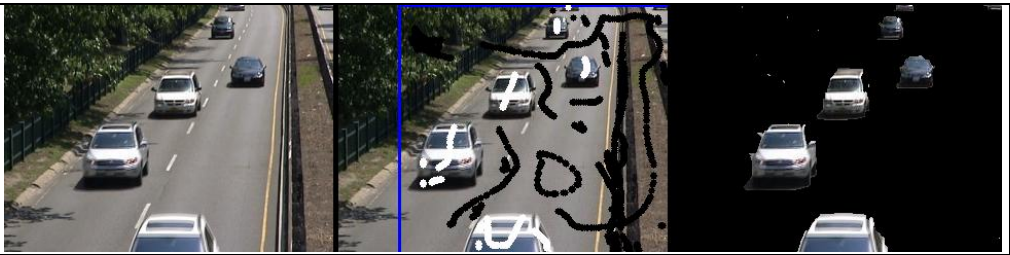
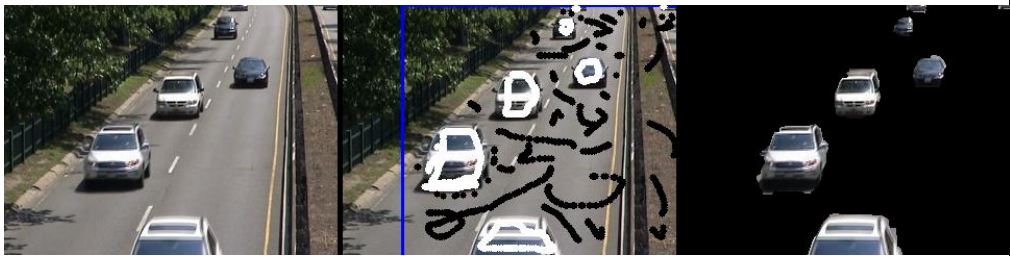
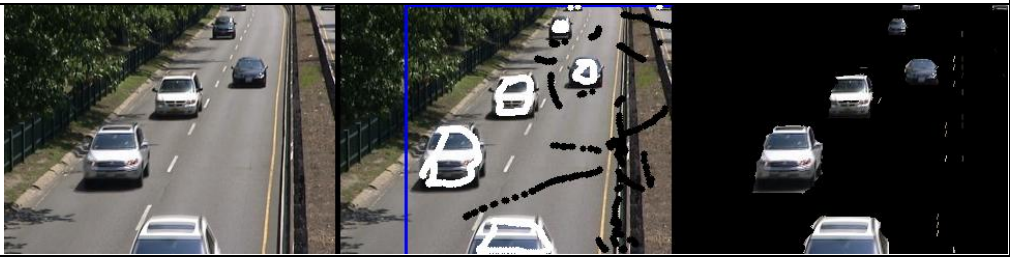
些边界分的不如其他两个方法，评估指标中，则是 L1 范数最好，切比雪夫效果次之，然后是欧氏距离，最后是余弦相似度。

第三个图片中，视觉效果和标注量差别不大，除了余弦相似度与其他图片在车轮处有所不同。评估指标中，则是余弦相似度最好，欧氏距离次之，切比雪夫略好于 L1 范数但是差距很小。

结论：从实际体验上来看，余弦相似度需要的标注比较多，而且效果也不太稳定，计算量大导致速度也是最慢的，不如欧氏距离；而 L1 范数和切比雪夫则不相上下，同时速度也是最快的，因为不需要平方等其他操作，效果也和欧氏距离类似甚至更好。

3.2 提取特征后再计算相似度结果对比




3.2.1 实验结果截图

不提取特征	
提取 LBP 特征	
提取 HOG 特征	

表格 7：图片一

不提取特征	
提取 LBP 特征	
提取 HOG 特征	

表格 8：图片二

不提取特征	
提取 LBP 特征	
提取 HOG 特征	

表格 9：图片三

3.2.2 实验结果数据

不 提 取 特 征	<pre> ----- Confusion matrix ----- Actual\Predicted Foreground Background Total Foreground 9505 2157 11662 Background 91 65047 65138 Total 9596 67204 76800 ----- Mean Pixel Accuracy(MPA): 90.68 % Accuracy: 97.07 % Precision: 99.05 % Recall: 81.5 % F1-socre: 89.42 % </pre>
提 取 LBP 特 征	<pre> ----- Confusion matrix ----- Actual\Predicted Foreground Background Total Foreground 9246 2416 11662 Background 194 64944 65138 Total 9440 67360 76800 ----- Mean Pixel Accuracy(MPA): 89.49 % Accuracy: 96.6 % Precision: 97.94 % Recall: 79.28 % F1-socre: 87.63 % </pre>
提 取 HOG 特 征	<pre> ----- Confusion matrix ----- Actual\Predicted Foreground Background Total Foreground 8770 2892 11662 Background 183 64955 65138 Total 8953 67847 76800 ----- Mean Pixel Accuracy(MPA): 87.46 % Accuracy: 96.0 % Precision: 97.96 % Recall: 75.2 % F1-socre: 85.08 % </pre>

表格 10: 图片一

不提 取特 征	<pre> ----- Confusion matrix ----- Actual\Predicted Foreground Background Total Foreground 8520 748 9268 Background 278 76854 77132 Total 8798 77602 86400 ----- Mean Pixel Accuracy(MPA): 95.78 % Accuracy: 98.81 % Precision: 96.84 % Recall: 91.93 % F1-socre: 94.32 % </pre>
提取 LBP 特征	<pre> ----- Confusion matrix ----- Actual\Predicted Foreground Background Total Foreground 8454 814 9268 Background 649 76483 77132 Total 9103 77297 86400 ----- Mean Pixel Accuracy(MPA): 95.19 % Accuracy: 98.31 % Precision: 92.87 % Recall: 91.22 % F1-socre: 92.04 % </pre>
提取 HO G 特征	<pre> ----- Confusion matrix ----- Actual\Predicted Foreground Background Total Foreground 8414 854 9268 Background 341 76791 77132 Total 8755 77645 86400 ----- Mean Pixel Accuracy(MPA): 95.17 % Accuracy: 98.62 % Precision: 96.11 % Recall: 90.79 % F1-socre: 93.37 % </pre>

表格 11: 图片二

不提 取特 征	<pre> ----- Confusion matrix ----- Actual\Predicted Foreground Background Total Foreground 39616 6804 46420 Background 18 291482 291500 Total 39634 298286 337920 ----- Mean Pixel Accuracy(MPA): 92.67 % Accuracy: 97.98 % Precision: 99.95 % Recall: 85.34 % F1-socre: 92.07 % </pre>
---------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

提取 LBP 特征	<pre> ----- Confusion matrix ----- Actual\Predicted Foreground Background Total Foreground 39991 6429 46420 Background 748 290752 291500 Total 40739 297181 337920 ----- Mean Pixel Accuracy(MPA): 92.95 % Accuracy: 97.88 % Precision: 98.16 % Recall: 86.15 % F1-socre: 91.76 % </pre>
	<pre> ----- Confusion matrix ----- Actual\Predicted Foreground Background Total Foreground 38493 7927 46420 Background 230 291270 291500 Total 38723 299197 337920 ----- Mean Pixel Accuracy(MPA): 91.42 % Accuracy: 97.59 % Precision: 99.41 % Recall: 82.92 % F1-socre: 90.42 % </pre>

表格 12：图片三

3.2.3 实验结果分析

第一张图片中，可以看到 HOG 特征的会出现很多缝隙，视觉效果不好，用户操作体验也不如欧氏距离，因为出现太多缝隙需要用户更多的交互；LBP 明显需要比不提取特征更多的标注量，视觉上无明显差别，但略差于不提取的。从指标上看，两种提取都不如不提取的效果好，而 HOG 效果差于 LBP 提取。

第二、第三张与第一个图片类似，会出现缝隙，或者分解出有缝隙；评估指标上，同样是 LBP 和 HOG 不如不提取特征的。

结论：LBP 和 HOG 提取特征后再计算相似度并没有提升，而且效果还略差于原先的。由于 LBP 提取的主要是边界纹理信息，而 HOG 主要提取的是梯度和方向信息，直接运用欧氏距离不如在图像像素上操作，说明提取特征导致像素信息丢失过多，效果变差了。

四、总结

本次实验比较麻烦，主要是需要用户交互，做对比实验时比较麻烦，因为有的方法会让效果变差，结果需要用户大量标注才能稍微弥补。最后不同的相似度衡量方法没有明显的差距，但是 L1 和切比雪夫距离表现良好，略好于欧氏距离，

余弦相似度则略差于欧氏距离；LBP 和 HOG 特征提取的效果并不好，都不如不提去特征，说明提取特征导致像素信息丢失过多，而我们需要的是像素间的相似度，丢失信息过多导致效果变差了。