参考csdn：https://blog.csdn.net/Ibelievesunshine/article/details/105043466

原图像为 I（H X W），待匹配图像为 T（h X w）。

1. **对于图像 I：，for j in range(H-h)：for i in range(W-w) 。在一次移动一像素的过程中，计算原图像的一部分 I[j：j+h，i：i+w] 与待匹配图像 T 的相似度 S。**

   S 最大或者最小的地方即为匹配到的位置。

S 的计算方法主要有 **SSD**(Sum of Squared Difference：误差平方和)、**SAD**(Sum of Absolute Differences：误差绝对值和)、**NCC**(Normalization Cross Correlation：归一化交叉相关)、**ZNCC**(Zero-mean Normalization Cross Correlation：零均值归一化交叉相关)，对于不同的方法，我们需要选择出 S 的最大值或者最小值，然后找到 S 的值对应的在原图像中的位置。

不同的模式匹配标准(不同的 S)

S = SSD(Sum of Squared Difference)，S 取最小的值。

$$S = \sum_{x=0}^{w} \sum_{y=0}^{h} [I(i+x, j+y) - T(x,y)]^2$$

SSD(误差平方和)最小 ↑

python实现核心代码：_v = np.sum((img[y:y+Ht, x:x+Wt] - temp) ** 2)

S = SAD(Sum of Absolute Differences)， S 取最小的值。

$$S = \sum_{x=0}^{w} \sum_{y=0}^{h} |I(i+x, j+y) - T(x,y)|$$

SAD(误差绝对值之和)最小 ↑

python实现核心代码：_v = np.sum(np.abs(img[y:y+Ht, x:x+Wt] - temp))

S = NCC(Normalization Cross Correlation)，S 取最大的值。求出两个图像的相似度，S 的范围为[-1,1]，S 对变化十分敏感。

$$S = \frac{\sum_{x=0}^{w} \sum_{y=0}^{h} I(i+x, j+y) \cdot T(x, y)}{\sqrt{\sum_{x=0}^{w} \sum_{y=0}^{h} I(i+x, j+y)^2} \cdot \sqrt{\sum_{x=0}^{w} \sum_{y=0}^{h} T(i, j)^2}}$$

NCC(归一化交叉相关)最大↑

python实现核心代码：_v = np.sum(img[y:y+Ht, x:x+Wt] * temp)

_v /= (np.sqrt(np.sum(img[y:y+Ht, x:x+Wt]**2)) * np.sqrt(np.sum(temp**2)))

S = ZNCC(Zero-mean Normalization Cross Correlation)，S 取最大的值。它比归一化交叉相关更加敏感。图像 I 的平均值为 m_i，图像 T 的平均值为 m_t。S 的范围为 [-1,1]。

$$S = \frac{\sum_{x=0}^{w} \sum_{y=0}^{h} [I(i+x, j+y) - m_i] \cdot [T(x, y) - m_t]}{\sqrt{\sum_{x=0}^{w} \sum_{y=0}^{h} [I(i+x, j+y) - m_i]^2} \cdot \sqrt{\sum_{x=0}^{w} \sum_{y=0}^{h} [T(x, y) - m_t]^2}}$$

ZNCC(零均值归一化交叉相关)最大↑

python实现核心代码：_v = np.sum((img[y:y+Ht, x:x+Wt]-mi) * (temp-mt))

_v /= (np.sqrt(np.sum((img[y:y+Ht, x:x+Wt]-mi)**2)) * np.sqrt(np.sum((temp-mt)**2)))

二值化之后四种方法效果都较好，但时间太长，现在如何切割主体部分是问题。

代码：https://github.com/Nicous20/Road_Pattern_Matching