

Project2 - Partical Filter

- 3140102282 王兴路 信息工程1403

Project2 - Partical Filter

feature_extract

compute_similarity

resample_step

maintain template

feature_extract

在特征提取的步骤中，我尝试过HOG，LBP以及关键点+描述器的方法。总体来说还是HOG速度最快，效果也较好。

compute_similarity

相似度可以用皮尔逊积矩相关系数度量(由于正相关、负相关都表示相关，因此加了绝对值)，一开始我使用for循环，处理一帧图片耗时0.52秒。

考虑到compute_similarity会被频繁地调用，我将相关性计算改为了矩阵操作，MATLAB可以并行计算，从而将耗时降到0.48秒（对参取值 `sz_I = [56, 56];`）。不过虽然compute_similarity被频繁调用，但是主要性能瓶颈在提取特征阶段，因此可能看起来速度上提升不算明显。

下面的矩阵操作版本是按照corr的等价标准分数均值估计实现的。

$$r = \frac{1}{n-1} \sum_{i=1}^n \left(\frac{X_i - \bar{X}}{\sigma_X} \right) \left(\frac{Y_i - \bar{Y}}{\sigma_Y} \right)$$

```
1. % Make each column zero-mean
2. Y = bsxfun( @minus, Y, mean( Y, 1) );
3. y = bsxfun( @minus, y, mean( y, 1) );
4. y= repmat(y,1,N2);
5. % L2 normalize each column
6. Y = bsxfun( @times, Y, 1./sqrt( sum( Y.^2, 1) ) );
7. y = bsxfun( @times, y, 1./sqrt( sum( y.^2, 1) ) );
8. % Take the dot product of the columns and then sum
9. s=sum( Y.*y, 1);
```

resample_step

在这一步中我遇到了一种比较麻烦的现象，叫做采样贫瘠（sample impoverishment）。我发现400(number of particles)乘以相应的权重，基本上都是1.2个。这是时候要是round的话，由于roundoff error，会导致采出来的particles数目变少。

一开始我采用的是老师所介绍的采样的方法，根据粒子的权重将[0,1]划分为400个区间。在for循环中，产生一个0-1的随机数，根据随机数的值选取相应区间的粒子。但是查阅资料后发现其实有一种叫做 **systematic sampling** 的等价的高效的采样方法。步骤就是：

- 计算权重的累积和，用这些累积和将[0,1]划分为400个区间。这一步是一样的。
- 而后只需要产生一个随机数即可！在区间[0,1/400]中，产生一个随机数。然后等间距地选取剩下的随机数。

maintain template

一开始我设计了很复杂的 **update_templates.m**，想通过类似排除异常点的方法维护多个模板。但是比较麻烦的是特征是一个高维向量，只知道相对的相关性，不知道绝对的排名。所以我实现出来的方法，总是维护着前五帧模板。

```
0.2500    0.2500    0.2500    0.2500

Elapsed time is 1.145994 seconds.

templates_weights =

0.2000    0.2000    0.2000    0.2000    0.2000

Elapsed time is 1.225461 seconds.

templates_weights =

0.2013    0.2013    0.1998    0.1990    0.1986

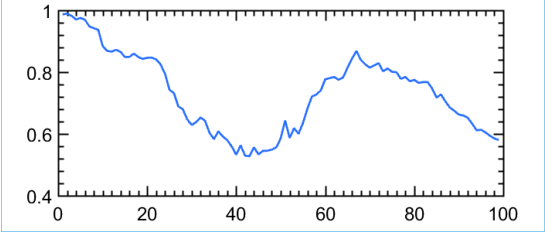
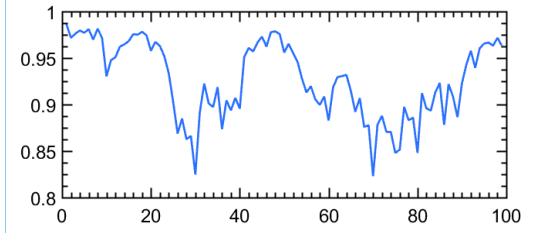
Elapsed time is 1.492435 seconds.

templates_weights =

0.2013    0.2013    0.1998    0.1990    0.1986
```

于是我尝试了指数平滑更新单个模板的方法。

采用不同的阈值平滑，相邻两帧相似度变化如下，才有0.7429的阈值，可以较好地更新模板，当人脸旋转的时候，框不会发生大的平移和缩放。

threshold	1	0.7429
相邻两帧相似 度变化		

然后最后我也发现Particle Filter其实有一些随机性，有的时候很容易更丢的47帧，也能较好地跟上。

