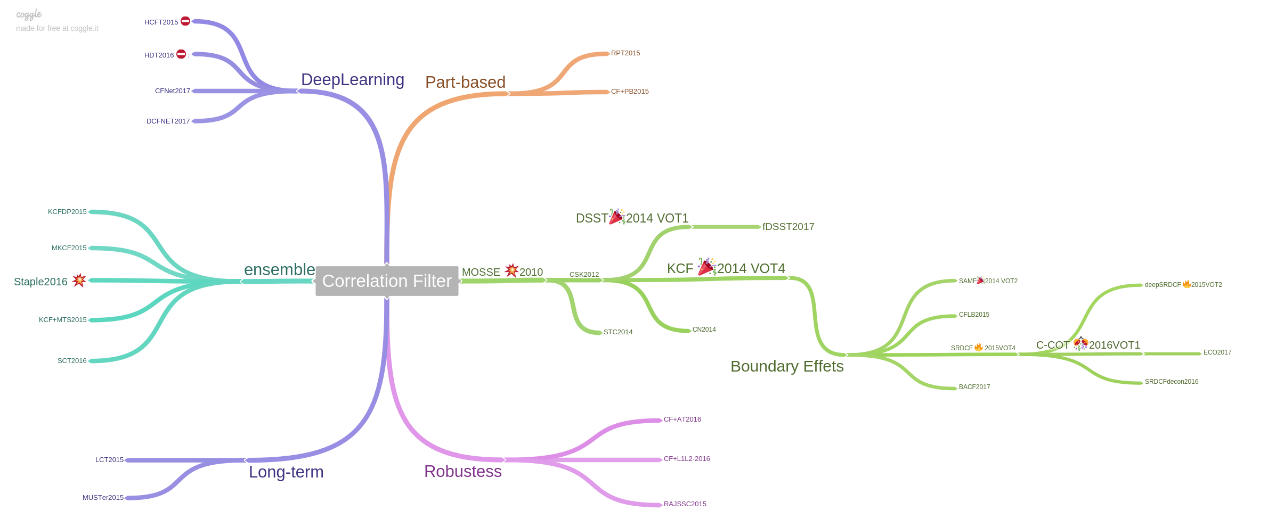
上海大学无人艇工程研究院

——环境感知组

# 视觉跟踪算法之相关滤波发展概述



图注：本文综述

Date：2017.04.21

注：以下**加粗**的作者为重点关注学者

## 版本更新记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日 期** | **更新人** | **主要更新内容描述** | **版本号** |
| 2017年04月21日 | 陈加宏 | 完成框架搭建，内容大纲填充 | V1.0.0 |
| 2017年04月24日 | 陈加宏 | 完善相关算法的理解及版面设计 | V1.0.1 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目 录

[视觉跟踪算法之相关滤波发展概述 1](#_Toc480815060)

[版本更新记录 1](#_Toc480815061)

[1、初见相关滤波与OTB 3](#_Toc480815062)

[2、大话目标跟踪—背景介绍 5](#_Toc480815063)

[3、NIUBILITY的相关滤波框架详解 10](#_Toc480815064)

[4、多尺度检测与VOT2014 16](#_Toc480815065)

[5、边界效应与VOT2015 20](#_Toc480815066)

[6、颜色保卫战与VOT2016 24](#_Toc480815067)

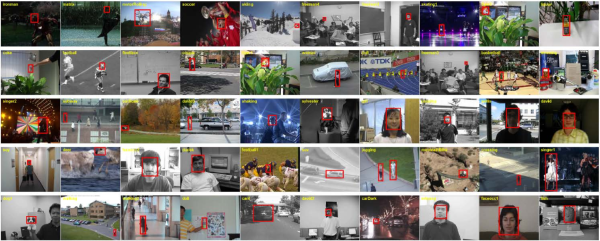
## 1、初见相关滤波与OTB

人生若只如，一切都从**2013年**开始说起，那一年的冬天和往常一样冷。如果你问别人近几年有什么比较niubility的跟踪算法，大部分人都会扔给你吴毅老师的论文，OTB50和OTB100(OTB50这里指OTB-2013，OTB100这里指OTB-2015，感谢指正)：

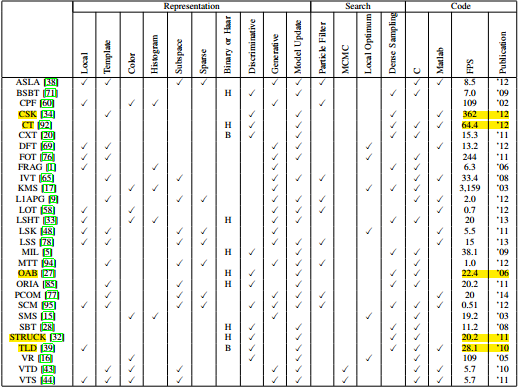
Wu Y, Lim J, Yang M H. Online object tracking: A benchmark [C]. CVPR, 2013.

Wu Y, Lim J, Yang M H. Object tracking benchmark [J]. TPAMI, 2015.

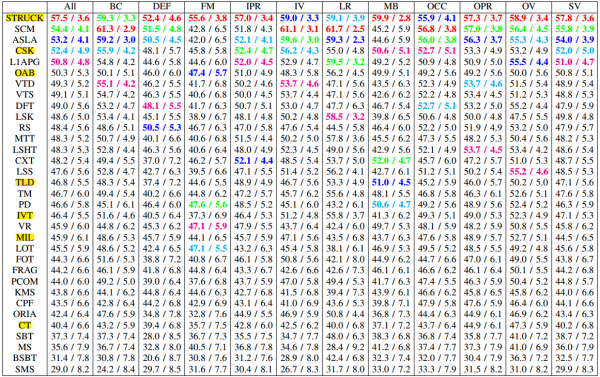
顶会转顶刊的顶级待遇，在加上引用量1200+200多，影响力不言而喻，已经是做tracking必须跑的数据库了，测试代码和序列都可以下载： Visual Tracker Benchmark，OTB50包括50个序列：



论文在数据库上对比了包括2012年及之前的29个顶尖的tracker，有大家比较熟悉的OAB, IVT, MIL, CT, TLD, Struck等，都是顶会转顶刊的神作，由于之前没有比较公认的数据库，论文都是自卖自夸，不知道到底哪个好用，所以这个database的意义非常重大，直接促进了跟踪算法的发展，后来又扩展为OTB100，有100个序列，我们这里参考OTB100的结果(标出了一些性能速度都比较著名的算法)：



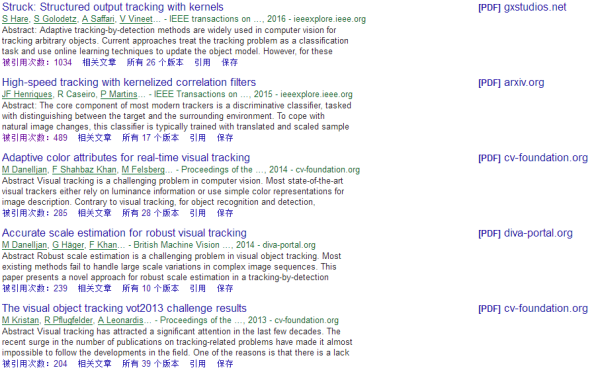
接下来再看看这29个顶尖tracker在OTB100上的实验结果：



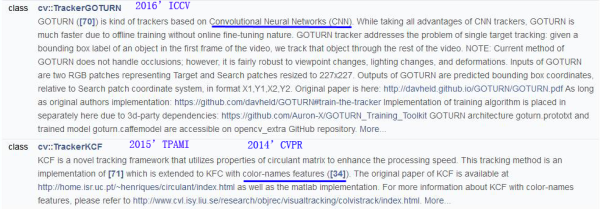
结论：平均来看**Struck, SCM, ASLA**等算法的性能比较高，排在前三。着重强调看一下CSK，第一次向世人展示了相关滤波的潜力，排第四还362FPS简直逆天了。请注意另一篇速度很快的经典算法CT(64fps)的排名(这段时间是压缩感知大热的时候，这里能看到很多稀疏相关算法，满满都是历史的痕迹)。如果对更早期的算法感兴趣，推荐另一篇经典的survey:

Yilmaz A, Javed O, Shah M. Object tracking: A survey [J]. CSUR, 2006.

自从**2012年AlexNet**问世以后，CV各个领域都有了很多变化，所以我猜你肯定还想知道**2013到2017年**发生了什么，抱歉我也不知道，不过我们可以肯定的是，2013年以后的论文确定以及必定都会引用OTB50这篇论文，**借助谷歌学术中的被引用次数功能**，得到如下结果：



这里仅列举几个排名靠前的，依次是Stuck转PAMI, 三大相关滤波方法KCF, CN, DSST, 和VOT竞赛，这里仅作示范，有兴趣可以去试试。（这么做的理论依据是：一篇论文，在它之前的工作可以看它的引用文献，之后的工作可以看谁引用了它；虽然引用量并不能说明什么，但**好的方法大家基本都会引用的**；之后还可以通过限定时间来查看某段时间的相关论文，如2016-2017就能找到最新的论文了，至于论文质量需要仔细甄别；其他方向的重要论文也可以这么用，顺藤摸瓜，然后你就知道大牛是哪几位，接着关注跟踪一下他们的工作(以上小技巧，如果你不知道，不谢！）这样我们就大致就知道目标跟踪领域的最大发展应该就是相关滤波无疑了，再往后还能看到相关滤波类算法有SAMF, LCT, HCF, SRDCF等等。引用量也与时间有关，建议分每年来看，这里就不贴图了(**相关滤波类新论文也可以通过引用KCF的论文去找**)。此外，最新版本opencv3.2除了TLD，也包括了几个很新的跟踪算法 OpenCV: Tracking API（ <http://www.docs.opencv.org/3.2.0/d9/df8/group__tracking.html> ）：



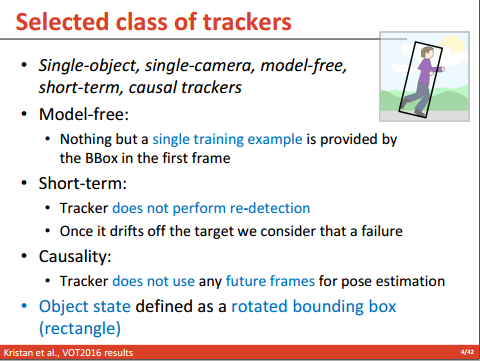
TrackerKCF接口实现了KCF和CN，影响力可见一斑，这一点就可以说明很多问题，还有个GOTURN是基于深度学习的方法，虽然在GPU上跑100FPS，**在VOT2014上接近DSST，在OTB上远不如KCF**，与同样发表在ECCV2016的SiamFC差太多了，而后者在GPU上也可以跑83FPS，opencv加入这个方法窃以为太激进了。

以上就是2012年以前目标跟踪算法的大致情况，之后就是相关滤波和深度学习的崛起！

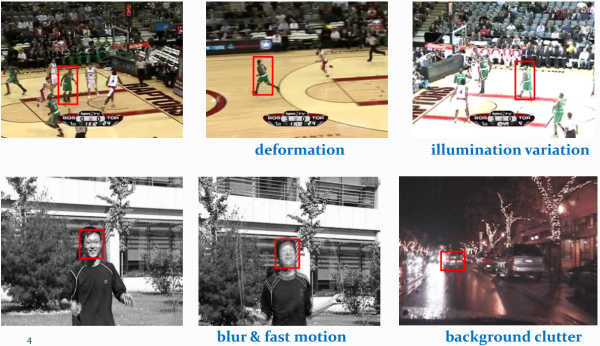
## 2、大话目标跟踪—背景介绍

在介绍项滤波之前，先总体简单介绍下目标跟踪。

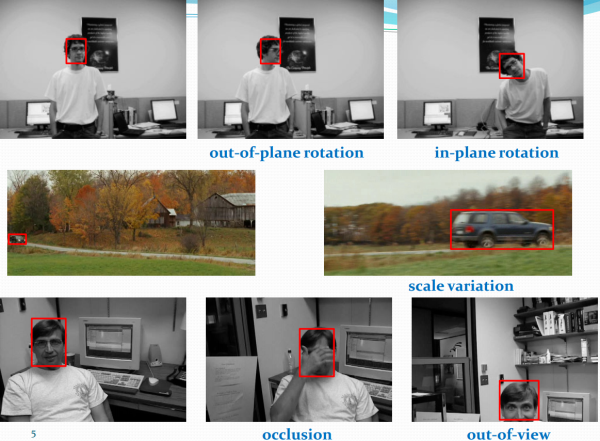
这里说的目标跟踪，是通用单目标跟踪，第一帧给个矩形框，**这个框在数据库里面是人工标注的**，在**实际情况下大多是检测算法的结果**，然后需要跟踪算法在后续帧紧跟住这个框，以下是VOT对跟踪算法的要求：



通常目标跟踪面临由几大难点(吴毅在VALSE视觉与学习青年学者研讨会的幻灯片：外观变形，光照变化，快速运动和运动模糊，背景相似干扰：



平面外旋转，平面内旋转，尺度变化，遮挡和出视野等情况：



正因为这些情况才让tracking变得很难，目前比较常用的数据库除了OTB，还有VOT竞赛数据库(堪比目标跟踪中的ImageNet，虽然差远了)，已经举办了四年，每年固定在ICCV或ECCV上有目标跟踪的workshop(盛宴)，VOT2015和VOT2016都包括60个序列，所有序列也是免费下载 VOT Challenge | Challenges：

Kristan M, Pflugfelder R, Leonardis A, et al. The visual object tracking vot2013 challenge results [C]// ICCV, 2013.

Kristan M, Pflugfelder R, Leonardis A, et al. The Visual Object Tracking vot2014 Challenge Results [C]// ECCV, 2014.

Kristan M, Matas J, Leonardis A, et al. The visual object tracking vot2015 challenge results [C]// ICCV, 2015.

Kristan M, Ales L, Jiri M, et al. The Visual Object Tracking VOT2016 Challenge Results [C]// ECCV, 2016.

**OTB和VOT区别**：**OTB包括25%的灰度序列，但VOT都是彩色序列，这也是造成很多颜色特征算法性能差异的原因；两个库的评价指标不一样**，具体请参考论文；VOT库的序列分辨率普遍较高，这一点后面分析会提到。对于一个tracker，如果两个库(最好是OTB100和VOT2016)都跑了且结果上佳，那性能肯定是非常优秀的(两个库调参你能调好，我服)，如果只跑了一个。 (仅供参考)我比较认可的是VOT2016，因为序列都是精细标注，且评价指标更加认可(毕竟是竞赛，评价指标发过TPAMI的) 最大差别在于：OTB是随机帧开始，或矩形框加随机干扰初始化去跑，作者说这样更加符合检测算法给的框框；**而VOT是第一帧初始化去跑，每次跟踪失败(预测框和标注框不重叠)时，5帧之后再次初始化，VOT以short-term为主，且认为跟踪检测应该在一起永不分离**，detecter会多次初始化tracker。至于哪个更好，看你的需求。

补充：OTB在2013年公开了，对之前的算法是不可见的，所以OTB论文的结果非常可靠，但对于2013以后的算法是透明的，有调参的可能性，尤其是那些只跑OTB，而且论文中有关键参数直接给出还精确到小数点后两位的算法，建议您先实测再评价(人心不古啊~被坑的多了)，但VOT竞赛的数据库是每年更新，还动不动就重新标注，动不动就改变评价指标，对当年算法是不可见且难度很大的，所以结果更可靠。

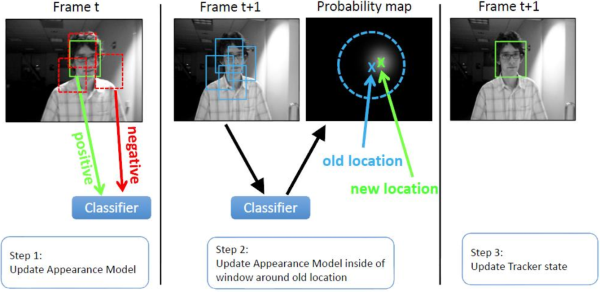
目标视觉跟踪(Visual Object Tracking)，大家比较公认分为两大类：生成(generative)模型方法和判别(discriminative)模型方法，目前比较流行的是判别类方法，也叫检测跟踪(tracking-by-detection)，为保持完整性，以下简述。

**生成类方法，在当前帧对目标区域建模，下一帧寻找与模型最相似的区域就是预测位置**。比较著名的有卡尔曼滤波，粒子滤波，mean-shift等。举个例子，从当前帧知道了目标区域80%是红色，20%是绿色，然后在下一帧，搜索算法就像无头苍蝇，到处去找最符合这个颜色比例的区域，推荐算法ASMS vojirt/asms：

Vojir T, Noskova J, Matas J. Robust scale-adaptive mean-shift for tracking [J]. Pattern Recognition Letters, 2014.

**VOT2015的第20名及官方推荐的实时算法，VOT2016的32名，平均帧率125FPS**，在经典mean-shift框架下加入了尺度估计，经典颜色直方图特征，加入了两个先验(尺度不剧变+可能偏最大)作为正则项，和反向尺度一致性检查，作者给了C++代码，在相关滤波和深度学习盛行的年代，还能看到mean-shift打榜，如此高的性价比实在不容易，实测性能还不错，如果您对生成类方法情有独钟，这个非常推荐您去试试。

判别类方法，OTB50里面的大部分方法都是这一类，经典套路图像特征+机器学习， 当前帧以目标区域为正样本，背景区域为负样本，机器学习训练分类器，下一帧用训练好的分类器找最优区域(马超在VALSE的slides)：



与生成类方法最大的区别，是分类器训练过程中用到了背景信息，这样分类器专注区分前景和背景，判别类方法普遍都比生成类好。举个例子，在训练时告诉tracker目标80%是红色，20%是绿色，还告诉它背景中有橘红色，要格外注意别搞错了，这样的分类器知道更多信息，效果也肯定更好。tracking-by-detection和检测算法非常相似，如经典行人检测用HOG+SVM，Struck用到了haar+structured output SVM，跟踪中为了尺度自适应也需要多尺度遍历搜索，区别仅在于跟踪算法对特征和在线机器学习的速度要求更高，检测范围和尺度更小而已。这点其实并不意外，大多数情况检测识别算法复杂度比较高不可能每帧都做，这时候用复杂度更低的跟踪算法就很合适了，只需要在跟踪失败(drift)或一定间隔以后再次检测去初始化tracker就可以了。其实我就想说，FPS才TMD是最重要的指标，经典判别类方法推荐**Struck和TLD**，都能实时性能还行，Struck是2012年之前最好的方法，TLD是经典long-term类跟踪的代表，即使效果差一点但思想非常值得借鉴：

Hare S, Golodetz S, Saffari A, et al. Struck: Structured output tracking with kernels [J]. IEEE TPAMI, 2016.

Kalal Z, Mikolajczyk K, Matas J. Tracking-learning-detection [J]. IEEE TPAMI, 2012.

判别类方法的最新发展就是**相关滤波类方法**，correlation filter简称CF，或discriminative correlation filter简称DCF(注意和后面KCF中的DCF方法区别，包括前面提到的那几个，后面要着重介绍)，和深度学习(Deep ConvNet based)类方法，因为深度学习类并不了解就不瞎推荐了，除了Winsty的几篇 Naiyan Wang - Home，还有VOT2015的冠军MDNet和VOT2016的冠军TCNN，速度方面比较突出的如**80FPS的SiamFC**和**100FPS的GOTURN**，注意是在**GPU**上。基于ResNet的**SiamFC-R**在VOT2016表现不错，很看好后续发展，如果有兴趣可以去VALSE听作者自己讲解，至于GOTURN，效果那么差，跑100FPS又有什么用呢?

Nam H, Han B. Learning multi-domain convolutional neural networks for visual tracking [C]// CVPR, 2016.

Nam H, Baek M, Han B. Modeling and propagating cnns in a tree structure for visual tracking. arXiv preprint arXiv:1608.07242, 2016.

Bertinetto L, Valmadre J, Henriques J F, et al. Fully-convolutional siamese networks for object tracking [C]// ECCV, 2016.

Held D, Thrun S, Savarese S. Learning to track at 100 fps with deep regression networks [C]// ECCV, 2016.

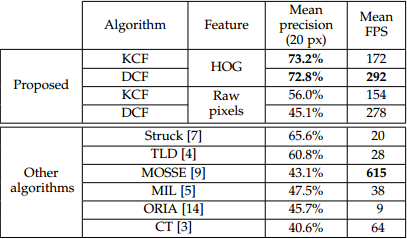
还有几篇国人大作遇到了不要惊讶：MEEM, TGPR, CLRST, MUSTer，都是2013年以后在OTB上性能比较高的，在VOT上也能看到他们的身影，但**速度是硬伤**。最后，深度学习**END2END**的强大威力在目标跟踪方向还远没有发挥出来，还没有和相关滤波类方法拉开多大差距，最后强力推荐**Qiang Wang**大神维护的资源benchmark\_results：大量顶级方法在OTB库上的性能对比，各种论文代码应有尽有，大神自己C++实现并开源的**CSK, KCF和DAT**，找不着路的同学请跟紧，别迷路**foolwood/benchmark\_results**，最近还开源了他自己最新论文DCFNet论文加源码，有兴趣的同学别错过。

再隆重推荐**贺辉大神**维护相关类方法的资源**HakaseH/CF\_benchmark\_results** ，详细分类和论文代码资源，走过路过别错过，是我见过的最详尽的相关滤波资源，可见大神非常之用心！

以上就是目标跟踪方向的简单背景介绍，主要为了说清楚目标跟踪最近发展，和相关滤波是哪棵树上哪个枝干上长出的果实，舞台搭好了，马上开始介绍相关滤波！

## 3、NIUBILITY的相关滤波框架详解

本文详细介绍经典高速相关滤波类跟踪算法CSK, KCF/DCF, CN。很多人最早了解CF，应该和我一样，都是被下面这张图吸引了：



这是KCF/DCF算法在OTB50上(那时候OTB100还没有发表)的实验结果，Precision和FPS碾压了OTB50上最好的Struck，看惯了勉强实时的Struck和TLD，飙到高速的KCF/DCF突然有点让人不敢相信，其实KCF/DCF就是在OTB上大放异彩的CSK的多通道特征改进算法。注意到那个超高速615FPS的MOSSE了吧，这是目标跟踪领域的第一篇相关滤波类方法，这其实是真正第一次显示了相关滤波的潜力。和KCF同一时期的还有个CN，在2014CVPR上引起剧烈反响的颜色特征方法，其实也是CSK的多通道颜色特征改进算法。从MOSSE(615)到 CSK(362) 再到 KCF(172) 和 CN(152)，速度虽然有点下降，但效果越来越好，而且始终保持在高速水平，接下来就详细分解这四个方法：

Bolme D S, Beveridge J R, Draper B A, et al. Visual object tracking using adaptive correlation filters [C]// CVPR, 2010.

Henriques J F, Caseiro R, Martins P, et al. Exploiting the circulant structure of tracking-by- detection with kernels [C]// ECCV, 2012.

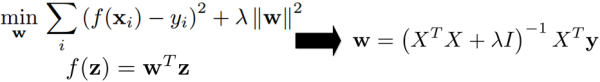
Henriques J F, Rui C, Martins P, et al. High-Speed Tracking with Kernelized Correlation Filters [J]. IEEE TPAMI, 2015.

Danelljan M, Shahbaz Khan F, Felsberg M, et al. Adaptive color attributes for real-time visual tracking [C]// CVPR, 2014.

由于CSK和KCF都是**Henriques J F(牛津大学)**大神先后两篇论文，且KCF/DCF是期刊内容比较充实，主要方法和CSK差别不大，因为单通道灰度特征容易可视化方便理解，我们就先从单通道灰度特征的没有kernel-trick的CSK开始了解相关滤波的算法原理和具体流程，然后再看看它和MOSSE的差别在哪里，最后再来看多通道特征扩展的KCF/DCF和CN。

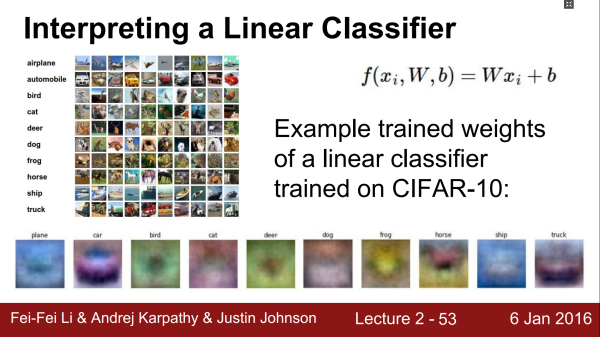
涉及算法本身是比较枯燥，KCF好多页都在推导，这里分部分来看这个算法。

首先是**岭回归(Ridge Regression)**。相关滤波也是判别类算法，选择了最简单的机器学习方法岭回归，虽然我一贯讨厌贴公式，但这里真的必须有几个(真的就几个)，岭回归和封闭解(closed-form solution )：



公式尽量少，所以具体参数含义和公式推导请看论文。岭回归这个大家应该非常熟悉，**最小二乘+二次正则项**，且最大优势是有封闭解可以快速优化。卷积操作与相关操作的区别仅在于是否将卷积核旋转180度，如果卷积核对称，这时候卷积与相关就是一回事了，现在CV中已经不做严格区分了，相关滤波的叫法来自信号处理。

为了对岭回归有深刻认识，借用CS231n(相信做CV的很多人都看过，必读精品)的一页PPT来解释线性分类器到底学到了什么，这样就容易理解相关滤波到底干了什么：



在cifar-10上训练的线性分类器，训练完成后的分类器参数W可视化以后，会看到10个包含了目标物体轮廓的模糊图像，这就是**分类模板(template)**，所以我们也可以这样类比岭回归：训练中我们得到了目标物体的一个模子，测试过程拿测试样本去逐个和模子对比，如果严丝合缝对上了，就会得到结果1；如果和模子比较相似，会得到接近1的结果；如果完全对不上，就会得到结果0。采用岭回归的相关滤波也是一样，所以相关滤波也可以叫做**模板类方法**，所用的特征叫做**模板类特征**。(注意区分**统计类特征**，如颜色直方图。你可能会想到Staple)

接下来就是**循环移位(Cyclic shifts)**。相关滤波的训练样本是通过循环移位产生的，理由1：**密集采样得到的样本与循环移位产生的样本很像，可以用循环移位来近似**。理由2：**卷积(相关)运算在傅里叶域对应点乘运算，可以减小计算量，而循环矩阵经过傅里叶变换会对角化，可以进一步减小计算量。**

训练标签是由高斯函数产生的连续值，相关滤波是回归而非分类，这点和以前的判别类方法区别较大。训练样本和标签如下图。原始样本是跟踪框加**1~1.5倍padding**得到的图像块，再乘一个余弦窗，训练样本集是原始样本通过循环移位产生，原始图像中心(红点)移位后对应高斯图位置的值就是这个移位样本的标签(蓝点)：



接下来将封闭解变换到傅里叶域，结合循环矩阵的特性将样本集进行化简，得到最终的封闭解：

用户上传的图片

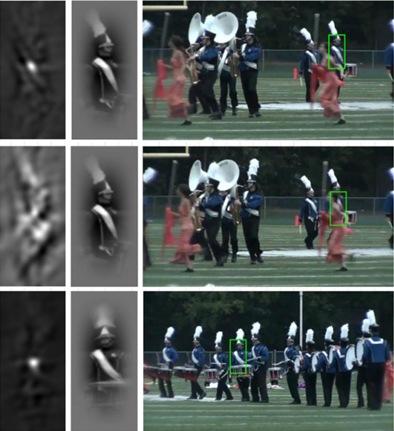
**其实循环移位的样本集是隐性的，并没有真正产生过，只是在推导过程中用到了，所以我们也不需要真正的内存空间去存储这些样本。**

快速检测。检测过程也是类似，用训练得到的分类器对检测区域图像块的每个循环移位样本做相关操作，得到一个响应值，所有移位样本的响应值会构成一幅响应图，与特征图的大小一致，**最大响应点就是跟踪目标的中心**。这个过程也可以在傅里叶域简化，而不用真正去循环移位产生检测样本并分类，计算高效节省内存：

用户上传的图片

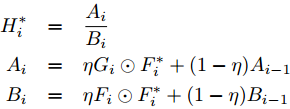
在举个简单例子，100\*100的检测块，目标中心位于边缘附近(20,20)位置，在检测过程中会持续循环移位并与滤波模板比较，只有当目标移位到中心时响应最大，这个最大值会写入响应图(20,20)位置。

此外，作者发现如果给岭回归加入kernel-trick，也可以推导出封闭解，推荐是**高斯核**。至此，CSK算法就成型了。总结一下，如果是100\*100的图像块，经过傅里叶变换和几个简单的元素级点乘计算，**事实上已经完成了10000个样本的训练和10000个样本的检测**，也就是说CSK的复杂度与图像块的像素点总数有关，padding大小会影响复杂度。循环移位生成的近似样本集结合FFT以近乎恐怖的方式降低了计算量，但是，这种近似以引入**边界效应**为代价，但是即使存在边界效应KCF/DCF依然碾压了Struck。接下来，再看几个可视化结果，依次是**某些帧的检测响应，模板权值和跟踪结果**：



首先看第二列的分类器参数，和前面的分析一致，可视化的模板权值就是跟踪目标的内容(模子，建模成功)。**相关响应在理想情况下呈现单峰，与训练时期望高斯图比较类似，有遮挡或在变形严重(模板对不上号)时会呈现多峰**。

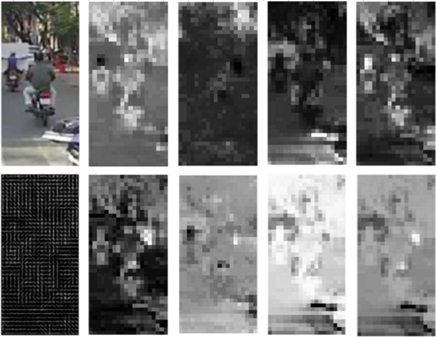
在MOSSE中，过程基本类似，第一帧初始化加了随机仿射，优化目标没有正则项，但最大区别还是训练和检测都没有加padding(主要区别)来扩展背景信息，仅建模了目标区域。**提出了一种跟踪置信度的评价指标峰值旁瓣比PSR**，更新方式是以固定学习率更新分类器参数，但形式略有区别，是滤波器的分子和分母分别更新，这种更新方式使用更加广泛：



然后，多通道特征扩展只需要频域通道响应求和，就这么简单。首先KCF/DCF将CSK中的单通道灰度特征扩展成**多通道的HOG特征**，采用**HOG+线性核的方法叫做DCF**，采用HOG+高斯核的方法叫做KCF，**依靠傅里叶变换的快速算法FFT和HOG的快速算法fHOG**，KCF/DCF成功起飞。作者提供了一份没有尺度更新的MATLAB代码，和三份C++代码，一份扩展了**LAB颜色特征**和简单的3尺度检测，一份扩展了**CN特征和尺度估计**，最后一份是opencv版本前面提到过，仅扩展了**CN特征没有尺度更新**。其实在这之前Hamed Kiani在MCCF中就尝试用多通道特征(SIFT, HOG)的MOSSE去做检测，证明HOG效果比较好，之所以会提到这个是因为这个名字后面还会出现：

Kiani Galoogahi H, Sim T, Lucey S. Multi-channel correlation filters [C]// ICCV, 2013.

同一时期Martin Danelljan大牛(林雪平大学，这个名字还会出现很多很多次)用多通道颜色特征Color Names(CN)去扩展CSK得到了不错的效果，算法也简称CN Coloring Visual Tracking。将RGB的3通道图像投影到11个颜色通道，分别对应英语中常用的语言颜色分类，分别是black, blue, brown, grey, green, orange, pink, purple, red, white, yellow (对比汉语中常用的语言颜色分类：赤橙黄绿青蓝紫+黑白灰，英语和汉语对颜色的认知还是略有差异的)，**并归一化得到10通道颜色特征**。作者还测试了多种颜色特征在相关滤波中的效果，发现CN最好，其次是**LAB**。下面是HOG特征和8个CN特征通道可视化的图像，随便画的随便看看：



接下来是CN的性能，作者选了41个彩色序列(25%灰度序列的OTB表现不出彩色特征方法的优势)进行比较，简单来说也是全面领先struck，10通道CN特征的FPS是78.9，压缩后两通道CN2特征的FPS是105，但这里需要强调一点，论文中的帧率FPS是和作者电脑的配置密切相关的，Martin Danelljan大神的电脑配置比较差算法普遍帧率低很多，这里我们以吴毅老师的电脑为参考，固定struck为20FPS，其他算法都换算到这个标准去比较，结果如下：**CSK(362FPS), KCF(172FPS), DCF(292FPS), CN(152FPS), CN2(202FPS)。（后面其他算法类似，如DSST是53.6FPS）**

用户上传的图片

**HOG是梯度特征，而CN是颜色特征，两者可以互补**，所以HOG+CN在近两年的跟踪算法中成为了**hand-craft特征标配**。最后，再回到KCF/DCF的实验结果，讨论两个问题：

**1. 为什么只用单通道灰度特征的KCF和用了多通道HOG特征的KCF速度差异很小？**

第一，HOG，作者用了HOG的快速算法fHOG，来自Piotr's Computer Vision Matlab Toolbox，C代码而且做了SSE优化。如对fHOG有疑问，请参考论文Object Detection with Discriminatively Trained Part Based Models第12页。

第二，HOG特征常用cell size是4，这就意味着，100\*100的图像，HOG特征图的维度只有25\*25，而Raw pixels是灰度图归一化，维度依然是100\*100，我们简单算一下： 结果也是相差不多，符合表格。

那你可能会想，如果下采样再多一点，复杂度就更低了，但这是以牺牲跟踪精度为代价的，再举个例子，如果图像块面积为200\*200，先下采样到100\*100，再提取HOG特征，分辨率降到了25\*25，这就意味着响应图的分辨率也是25\*25，也就实说，响应图每位移1个像素，原始图像中跟踪框要移动8个像素，这样就降低了跟踪精度。在精度要求不高时，你完全可以稍微牺牲下精度提高帧率。

**2. HOG特征的KCF和DCF哪个更好？**

大部分人都会认为KCF效果超过DCF，而且各属性的准确度都在DCF之上，然而，如果换个角度来看，以DCF为基准，再来看加了kernel-trick的KCF，mean precision仅提高了0.4%，而FPS下降了41%，这么看是不是挺惊讶的呢？除了图像块像素总数，KCF的复杂度还主要和kernel-trick相关。所以，下文中的CF方法如果没有kernel-trick，就简称基于DCF，如果加了kernel-trick，就简称基于KCF(请注意区分其他方法有没有加核)。当然这里的CN也有kernel-trick，但请注意，这是**Martin Danelljan大神第一次使用kernel-trick，也是最后一次。**

可能会有这样的疑问，kernel-trick这么强大的东西，怎么才提高这么点？这里就不得不提到Winsty的另一篇大作：

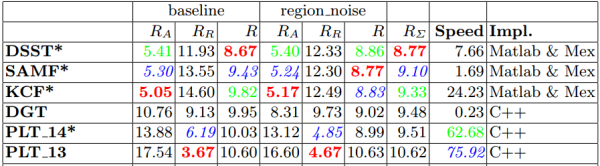
Wang N, Shi J, Yeung D Y, et al. Understanding and diagnosing visual tracking systems [C]// ICCV, 2015.

**一句话总结，别看那些五花八门的机器学习方法，那都是虚的，目标跟踪算法中特征才是最重要的。**(若有曲解，以论文为准)所以对于那些仅简单修改机器学习方法而不关注如何减小边界效应的相关滤波，也许可能大概会有效果，但在这里我一律不做评价讨论，我更加关心那些速度比较快或者对**边界效应处理**的比较有效的方法。

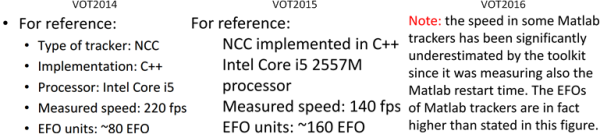
以上就是前三个首先推荐的高速算法，CSK, KCF/DCF和CN，区别主要在于有没有Kernel trick和用什么特征（灰度像素，fHOG，Color Names）！当然任性组合也是没有问题的。

## 4、多尺度检测与VOT2014

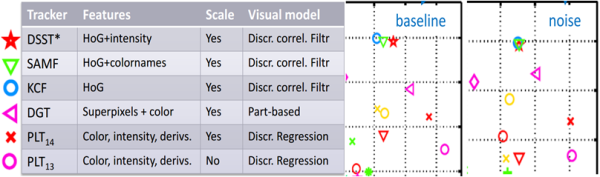
这篇文章主要介绍相关滤波中的多尺度检测和更新。光阴似水如白驹过隙，已经距离OTB2013发表过去一年了，VOT2014竞赛 VOT2014 Benchmark 。这一年有**25个精挑细选的序列，38个tracker**，那时候深度学习的战火还没有烧到tracking，所以也只能CF独霸一方了，下面是前几名的具体情况：



前三名都是相关滤波类(CF)方法，第三名的KCF已经很熟悉了，这里稍微有点区别就是**加了多尺度检测和子像素峰值估计**，再加上VOT序列的分辨率比较高(检测更新图像块的分辨率比较高)，导致KCF的速度只有24.23(EFO换算66.6FPS)。这里的speed指的的**EFO(Equivalent Filter Operations)**，在VOT2015和VOT2016里面也用这个参数衡量算法速度，每年EFO与帧率的关系这里一次性列出来供参考，VOT2016还强调MATLAB实现的Tracker实际速度要比EFO高：



其实前三名除了特征略有差异，核心都是扩展了多尺度检测，概要如下：



**尺度变化是跟踪中比较基本和常见的问题**，前面介绍的三个算法都没有尺度更新，如果目标缩小，滤波器就会学习到大量背景信息，如果目标扩大，滤波器就跟着目标局部纹理走了，这两种情况都很可能出现非预期的结果，导致漂移和失败。

首先介绍SAMF，来自浙大的工作，基于KCF，特征是HOG+CN：

Li Y .A scale adaptive kernel correlation filter tracker with feature integration ECCV, 2014.

可以看出这篇论文就在KCF和CN出来后不久，也是**第一次HOG和CN结合**，SAMF的多尺度检测方法用作者的原话是**straightforward**(论文里出现这个词，隐含的意思就是“这么做就行了，没啥创新”)，类似检测算法里面常用的多尺度检测方法，1.005间隔的7个尺度缩放图像分别检测，然后取7个响应图的最大值点，为了看清我把间隔扩大到1.05的5个尺度：



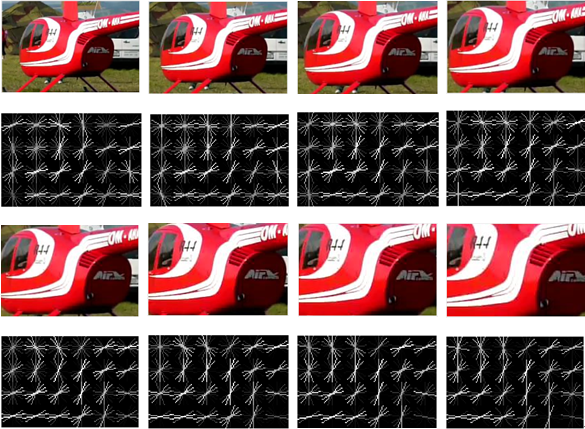
最佳尺度的目标点肯定有最大响应值，因此这种方法可以同时检测目标中心变化尺度变化，SAMF的速度是**7FPS**，因为速度比较慢，就介绍这么多。VOT2014中的KCF也采用这种尺度更新方法，步长1.05的3个尺度检测和更新，换算速度是66FPS，**可见减少尺度增加步长速度会快很多。**

接下来是Martin Danelljan的DSST Accurate scale estimation for visual tracking，基于DCF，特征只有HOG，但排名却在SAMF之前，得益于**更加简单精确的尺度方法**：

Danelljan M, Häger G, Khan F, et al. Accurate scale estimation for robust visual tracking [C]// BMVC, 2014.

Danelljan M, Hager G, Khan F S, et al. Discriminative Scale Space Tracking [J]. IEEE TPAMI, 2017.

**将目标跟踪看成目标中心平移和目标尺度变化两个独立问题**，首先用HOG特征的DCF训练平移相关滤波，负责检测目标中心平移，然后用HOG特征的MOSSE(注意区别，不加padding)训练另一个尺度相关滤波，负责检测目标尺度变化，(标配)步长1.02的33个尺度，选8个示例：



尺度滤波器仅需要检测哪个尺度最匹配而无须关心平移情况，所以DSST将特征压缩成一维(没有循环移位)，尺度检测的响应图也是一维的(1\*33)，同时为了减小计算量，DSST将尺度检测图像块全部缩放到512像素以内，所以DSST能精细检测33个尺度变化同时保持可观的帧率(论文中是**24FPS**)。更新阶段两个滤波器都以固定学习率逐帧更新。DSST转PAMI后的快速改进版本fDSST虽然性能略有下降，帧率达到了**54.3FPS**(标准化后是**114FPS**)。

以上就是两种最常用的多尺度检测方法，最后，讨论两个问题：

**DSST和SAMF所采用的尺度检测方法哪个更好？**

首先给大家讲个笑话：Martin Danelljan大神提出DSST之后，他的后续论文就再没有用过。虽然SAMF和DSST都可以跟上普通的目标尺度变化，但SAMF只有7个尺度比较粗，而DSST有33个尺度比较精细准确；DSST先检测最佳平移再检测最佳尺度，是分布最优，而SAMF是平移尺度一起检测，是平移和尺度同时最优，而往往局部最优和全局最优是不一样的；DSST看成两个问题可以采用不同的方法和特征更加灵活，但需要额外训练一个滤波器，每帧尺度检测需要采样33个图像块，分别计算特征，加窗，FFT，尺度滤波器还要额外训练；SAMF只需要一个滤波器，不需要额外训练和存储，每个尺度检测就一次提特征和FFT，但在图像块较大时计算量比DSST高。所以尺度检测DSST并不总是比SAMF好，其实，在VOT2015和VOT2016上SAMF都是超过DSST的，当然这主要是因为特征更好，但至少说明尺度方法不差。虽然DSST比SAMF更具创新度，但SAMF也是很优秀的方法。(记得高中数学老师说过，“一个人一生能有一点点创新就非常了不起了”，所以我们才会看到那么那么多灌水论文，创新真的太难了，那些虽然创新力不够但踏踏实实有效果，敢公开源码经得起验证的论文同样值得钦佩)

**DSST对尺度数量非常敏感吗？一定要33个尺度吗？**

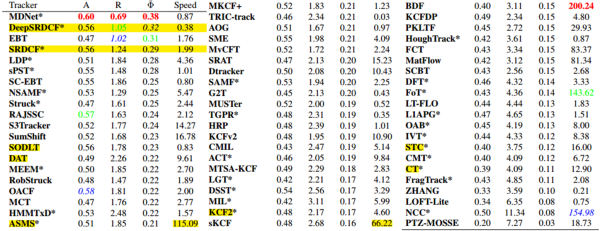
如果你认真跑过实验，就会发现DSST标配的33个尺度非常敏感，轻易降低尺度数量(如21个)，即使增加相应步长，尺度滤波器也会完全跟不上尺度变化。关于这一点我想到的可能解释是，**训练尺度滤波器用的是一维样本，而且没有循环移位，这就意味着一次训练更新只有33个样本，如果降低样本数量，会造成训练不足，分类器判别力严重下降，**不像平移滤波器有非常多的移位样本。总之，请不要轻易尝试大幅降低尺度数量，如果非要用尺度滤波器标配33和1.02就好。

以上就是两种推荐的尺度检测方法，以后通过多尺度缩放检测的方法简称为**类似SAMF的多尺度，而专门训练尺度滤波器检测尺度的方法简称为类似DSST的多尺度**。

如果您对尺度的要求不高而更看重速度，SAMF只要3个尺度就可以粗略跟上尺度变化(比如VOT2014中的KCF)；如果您需要精确尺度跟踪，则推荐DSST但速度稍慢，且DSST的尺度数量不能轻易减少；如果尺度变化不明显则不推荐使用。

## 5、边界效应与VOT2015

光阴荏苒岁月如流时间到了2015年，VOT2015竞赛 VOT2015 Challenge | Home 如期而至，这一年有**60个精挑细选的序列，62个tracker，**最大看点是深度学习开始进击tracking，MDNet直接拿下当年的冠军，而结合深度特征的相关滤波方法DeepSRDCF是第二名，主要**解决边界效应的SRDCF仅HOG特征排在第四**：



随着VOT竞赛的影响力扩大，举办方也是用心良苦，经典的和顶尖的齐聚一堂，百家争鸣，多达62个tracker皇城PK，华山论剑。除了前面介绍的深度学习和相关滤波，还有结合**object proposals(类物体区域检测)的EBT**排第三，**Mean-Shift类颜色算法ASMS是推荐实时算法**，还请注意另一个颜色算法DAT，而第9的那个Struck已经不是OTB中的那个了。除此之外，你还能看到各种博文力推主打的张开华老师的CT(12'ECCV)和STC(14 ECCV)，分别排倒数第6和第8。张开华老师我还是很尊重的，这里就事论事分析结果，不涉及张老师的其他论文方法。CT在OTB上排倒数第四希望您还记得，而同为12'ECCV的CSK也排第四，不过是正着数的，而STC和同为14'ECCV的SAMF(排28)差了很多，所以有同学反映实测CT和STC没有论文中说的那么好，答案就在这里，值得参考但我不推荐深入学习和研究，更不推荐您用在产品中。如果您不认可我的观点，至少这个结果也可以证明VOT竞赛非常激烈，就连CT和STC也只能排倒数。(冒着被打的风险说一句大实话，如果有人向你推荐CT或STC，要么他根本就没做过，甚至没跑过目标跟踪算法，信口胡说的，要么，他就是居心叵测，您最好仔细想想是不是欠他钱没还)

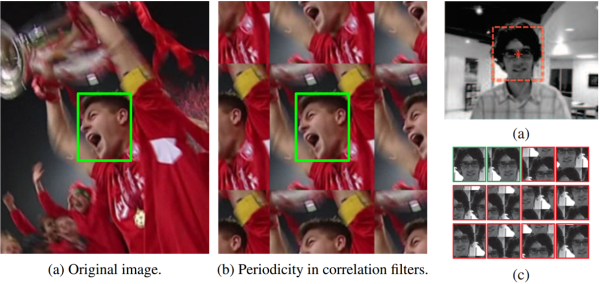
接下来主要介绍**相关滤波的边界效应及解决方案**，仅适合需要改进相关滤波类算法或者需要对相关滤波算法有更高要求的同学看，写出来是为了全面，而且后续改进都是以牺牲帧率为代价的，效果虽好但很多算法都无法实时了，任何算法都有优缺点，适合你的才是最好的。

总体来说，**相关滤波类方法对快速变形和快速运动情况的跟踪效果不好**。

快速变形主要因为CF是模板类方法。容易跟丢这个比较好理解，前面分析了相关滤波是模板类方法，如果目标快速变形，那基于HOG的梯度模板肯定就跟不上了，如果快速变色，那基于CN的颜色模板肯定也就跟不上了。这个还和模型更新策略与更新速度有关，固定学习率的线性加权更新，如果学习率太大，部分或短暂遮挡和任何检测不准确，模型就会学习到背景信息，积累到一定程度模型跟着背景私奔了，一去不复返。如果学习率太小，目标已经变形了而模板还是那个模板，就会变得不认识目标。(举个例子，多年不见的同学，你很可能就认不出了，而经常见面的同学，即使变化很大你也认识，因为常见的同学在你大脑里面的模型在持续更新，而多年不见就是很久不更新)

**快速运动主要是边界效应(Boundary Effets)**，而且边界效应产生的错误样本会造成分类器判别力不够强，下面分训练阶段和检测阶段分别讨论。

1、训练阶段，合成样本降低了判别能力。如果不加余弦窗，那么移位样本是长这样的：



除了那个最原始样本，其他样本都是“合成”的，100\*100的图像块，只有1/10000的样本是真实的，这样的样本集根本不能拿来训练。如果加了余弦窗，由于图像边缘像素值都是0，循环移位过程中只要目标保持完整那这个样本就是合理的，只有目标中心接近边缘时，目标跨越边界的那些样本是错误的，这样虽不真实但合理的样本数量增加到了大约2/3(padding= 1)，即使这样仍然有1/3(3000/10000)的样本是不合理的，这些样本会降低分类器的判别能力。再者，加余弦窗不是免费的，余弦窗将图像块的边缘区域像素全部变成0，大量过滤掉分类器本来非常需要学习的背景信息，原本训练时判别器能看到的背景信息就非常有限，我们还加了个余弦窗挡住了背景，这样进一步降低了分类器的判别力。

2、检测阶段，相关滤波对快速运动的目标检测比较乏力。相关滤波训练的图像块和检测的图像块大小必须是一样的，这就是说你训练了一个100\*100的滤波器，那你也只能检测100\*100的区域，如果打算通过加更大的padding来扩展检测区域，那样除了扩展了复杂度，并不会有什么好处。目标运动可能是目标自身移动，或摄像机移动，按照目标在检测区域的位置分四种情况来看：

1、如果目标在中心附近，检测准确且成功。

2、如果目标移动到了边界附近但还没有出边界，加了余弦窗以后，部分目标像素会被过滤掉，这时候就没法保证这里的响应是全局最大的，而且，这时候的检测样本和训练过程中的那些不合理样本很像，所以很可能会失败。

3、如果目标的一部分已经移出了这个区域，而我们还要加余弦窗，很可能就过滤掉了仅存的目标像素，检测失败。

4、如果整个目标已经位移出了这个区域，那肯定就检测失败了。

以上就是边界效应(Boundary Effets)，接下来简单介绍解决边界效应的方法，再次强调这些方法速度比较慢，**相关滤波傲视群雄的高速已经不见了，但换来了可以匹敌深度学习方法的性能。**

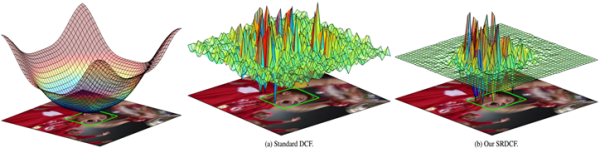
首先是VOT2015中最亮眼的相关滤波方法，Martin Danelljan大牛的SRDCF，主要思路：**既然边界效应发生在边界附近，那就忽略所有移位样本的边界部分像素，或者说让边界附近滤波器系数为接近零：**

Danelljan M, Hager G, Shahbaz Khan F, et al. Learning spatially regularized correlation filters for visual tracking [C]// ICCV. 2015.

SRDCF基于DCF，类SAMF多尺度，采用更大的检测区域(padding = 4)，同时加入空域正则化，惩罚边界区域的滤波器系数，优化目标如下：

用户上传的图片

空域正则化函数w，传统DCF响应和SRDCF响应对比图如下(来自论文)：



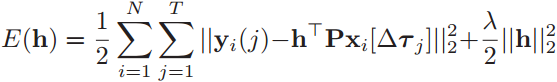
边界效应发生在原始样本目标中心循环移位到边缘附近时，所以越靠近边缘区域，正则化系数越大，惩罚并抑制这里的滤波器系数(有意忽略边界附近的样本值)。注意区分余弦窗是加在原始图像块上的，之后进行移位产生样本集；而空域正则化是加在样本集中每个移位样本上的。加正则项这种方法属于伤筋动骨类方法，破坏了DCF的封闭解，所以SRDCF采用高斯-塞德尔方法迭代求解最优化，这也是是导致SRDCF比较慢的原因。padding=4检测和训练图像块更大，为了减小计算量SRDCF控制滤波器的大小不超过50\*50，虽然速度只有5FPS，但实测对快速运动的提升非常明显。

另一种方法是Hamed Kiani(**前面提到过他MCCF，这位大牛专注改进MOSSE，好像世界的尘嚣和他没有一点关系**)提出的基于灰度特征MOSSE的CFLM Correlation Filters with Limited Boundaries 和基于HOG特征的BACF，**主要思路是采用较大尺寸检测图像块和较小尺寸滤波器来提高真实样本的比例，或者说滤波器填充0以保持和检测图像一样大**：

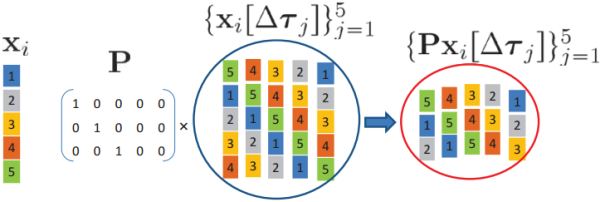
Kiani Galoogahi H, Sim T, Lucey S. Correlation filters with limited boundaries [C]// CVPR, 2015.

Kiani Galoogahi H,, Fagg A, Lucey S. Learning Background-Aware Correlation Filters for Visual Tracking [C]// arXiv preprint arXiv:1703.04590, 2017.

首先采用更大的检测和训练图像块，然后给循环移位样本左乘一个0和1组成的掩膜矩阵P，以使真实训练样本的比例增加，优化目标如下：



掩膜矩阵其实就是从大图像块的循环移位样本中裁切小图像块，一维样本为例来说明：



5\*1的原始样本，经过循环移位产生5个新样本，只有第五个是真实的，其他都是“合成”的，真实样本数/总样本数 = 1/5；给所有移位样本左乘一个掩膜矩阵P后，虽然样本尺寸变小为3\*1，但第三、四、五都是真实样本，真实样本数/总样本数 = 3/5，只有第一、二样本有边界效应，相当于裁切出了前三行。这就是从大尺寸样本的循环移位样本集中用掩膜矩阵裁切出真实的小尺寸样本，提升真实样本的比例，而且这样的掩膜矩阵不止一个，P的循环移位矩阵也有类似效果，即P指定裁切哪个区域，而裁切不同区域的图像块都能提升这个真实样本比例，而且原始样本越大，裁切后样本越小，则真实样本比例越高。

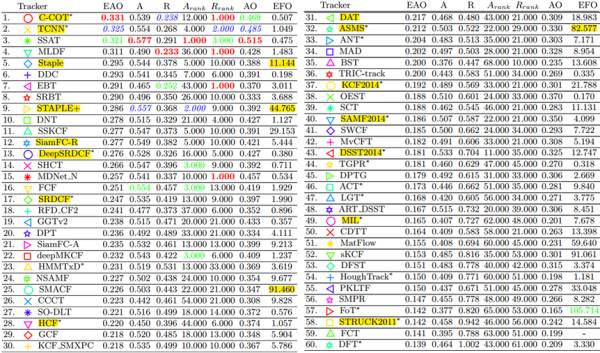
CFLM和BACF都属于**伤筋动骨类方法，没有闭合解**，采用**Alternating Direction Method of Multipliers(ADMM)迭代求解最优化**。按照论文，仅灰度特征的CFLM是基于MOSSE的，加了2倍padding用于裁切出原始目标大小的图像块(即相当于裁切后不加padding)，虽然性能真的很差，但速度有87.1FPS，而多通道HOG特征的BACF，加入类SAMF多尺度，性能全面超过SRDCF，在OTB50上竟然超过了VOT2016排第一名的C-COT，速度竟然有35FPS，感觉比SRDCF方法更有潜力，具体细节等作者公开代码实测，推荐先实测如果比较好再使用。

其实可以看出来这两个解决方案挺像的，都出现在2015年，都是用更大的检测区域，训练作用域比较小的相关滤波器，要么从中心到边缘逐渐衰减到0，要么在滤波器以外区域填充0。

在VOT2015中还有排在第二名的另一篇相关滤波方法，结合深度特征的DeepSRDCF，因为深度特征都非常慢，在CPU上别说高速，实时都到不了，所以放在最后面，和HCF、C-COT、最新ECO一起讨论。

## 6、颜色保卫战与VOT2016

弹指一挥间就到了去年，VOT2016竞赛 VOT2016 Challenge | Home，**依然是VOT2015那60个序列，不过这次做了重新标注更加公平合理**，今年有**70位参赛选手(tracker)**，意料之中深度学习已经雄霸天下了，**8个纯CNN方法和6个结合深度特征的CF方法大都名列前茅**，还有一片的CF方法，最最最重要的是，良心举办方竟然公开了他们能拿到的38个tracker，部分tracker代码和主页，下载地址：VOT2016 Challenge | Trackers，注意部分是下载链接，部分是源码压缩包，部分源码是二进制文件，好不好用一试便知，方便对比和研究，需要的赶快去试试。马上来看竞赛结果(这里仅列举前60个)：



标出来了前面介绍过的或比较重要的方法，结合多层深度特征的相关滤波C-COT排第一名，而CNN方法TCNN是VOT2016的冠军，作者和MDNet是同一人，纯颜色方法DAT和ASMS都在中等水平，其他tracker的情况请参考VOT竞赛论文(张开华老师的STC，排名很稳定在倒数第6，依然说明竞赛很激烈)。再来看速度，SMACF没有公开代码，ASMS依然那么快，排在前10的方法中也有两个速度比较快，**分别是排第5的Staple**，和其改进算法排第9的**STAPLE+**，而且STAPLE+是今年的推荐实时算法。首先恭喜Luca Bertinetto大牛的SiamFC和Staple都表现非常不错，然后在为大牛默哀三分钟(VOT2016的paper原文)：

This was particularly obvious in case of SiamFC trackers, which runs orders higher than realtime (albeit on GPU), and Staple, which is realtime, but are incorrectly among the non-realtime trackers.

VOT2016竟然发生了乌龙事件，Staple在论文中是80FPS，怎么EFO在这里只有11？幸好Staple和STAPLE+都公开了代码，上面下载地址都有，有兴趣您可以去实测下，虽然我电脑不如Luca Bertinetto大牛，但Staple我也能跑60-70FPS，而更可笑的是，STAPLE+比Staple慢了大约7-8倍，竟然EFO高出4倍，到底怎么回事呢？

首先看Staple的代码，如果您直接下载Staple并设置params.visualization = 1来跑，Staple调用**Computer Vision System Toolbox**来显示序列图像，而恰好如果您没有这个工具箱，默认每帧都会用imshow(im)来显示图像所以非常非常慢，而设置params.visualization = 0就跑的飞快(作者你是孙猴子派来的逗逼吗)，建议您将显示图像部分代码替换成DSST中对应部分代码就可以正常跑和显示了。

再来看STAPLE+的代码，改进包括额外从颜色概率图中提取HOG特征，特征增加到56通道(Staple是28通道)，平移检测额外加入了大位移光流运动估计的响应，所以才会这么慢，而且肯定会这么慢。

所以很大可能是VOT举办方把Staple和STAPLE+的EFO弄反了，VOT2016的实时推荐算法应该是排第5的Staple，相关滤波结合颜色方法，没有深度特征更没有CNN，跑80FPS还能排在第五，这就是接下来主要介绍的，2016年最NIUBILITY的目标跟踪算法之一Staple。

颜色特征，在目标跟踪中颜色是个非常重要的特征，不管多少个人在一起，只要目标穿不用颜色的一幅就非常明显。前面介绍过2014年CVPR的CN是相关滤波框架下的模板颜色方法，这里隆重介绍统计颜色特征方法DAT可能是由于相关滤波中HOG的效果太好了，所以作者展开了颜色保卫战：

Possegger H, Mauthner T, Bischof H. In defense of color-based model-free tracking [C]// CVPR, 2015.

DAT统计前景目标和背景区域的颜色直方图并归一化，这就是前景和背景的颜色概率模型，以后每帧线性插值更新颜色直方图；检测时对检测区域每个像素，根据其像素颜色值在前景和背景中的概率，贝叶斯方法判别这个像素属于前景的概率，得到像素级颜色概率图，在加上高斯权值函数抑制边缘相似颜色物体，就能得到目标的区域了：

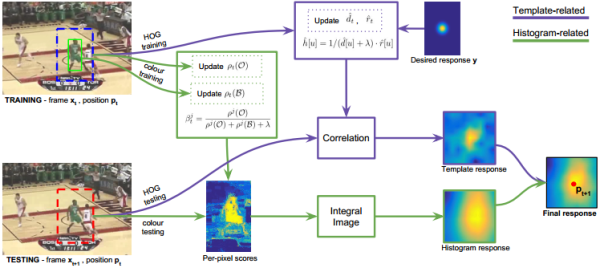


具体定位和尺度方法请参考论文。DAT在VOT上都略高于ASMS，帧率15FPS，这就是颜色保卫战DAT。

如果要用一句话介绍Luca Bertinetto(牛津大学)的Staple Staple tracker，那就是把模板特征方法DSST(基于DCF)和统计特征方法DAT结合：

Bertinetto L, Valmadre J, Golodetz S, et al. Staple: Complementary Learners for Real-Time Tracking [C]// CVPR, 2016.

前面分析了相关滤波模板类特征(HOG)对快速变形和快速运动效果不好，但对运动模糊光照变化等情况比较好；而颜色统计特征(颜色直方图)对变形不敏感，而且不属于相关滤波框架没有边界效应，快速运动当然也是没问题的，但对光照变化和背景相似颜色不好。综上，这两类方法可以互补，也就是说DSST和DAT可以互补结合：



简单吗？非常不简单，两个框架的算法高效无缝结合，25FPS的DSST和15FPS的DAT，而结合后速度竟然达到了80FPS。DSST框架把跟踪划分为两个问题，即平移检测和尺度检测，DAT就加在平移检测部分，相关滤波有一个响应图，像素级前景概率也有一个响应图，两个响应图线性加权得到最终响应图，其他部分与DSST类似，平移滤波器、尺度滤波器和颜色概率模型都以固定学习率线性加权更新。

**既然是颜色特征，DSST+DAT换成DSST用HOG+CN一样吗？**

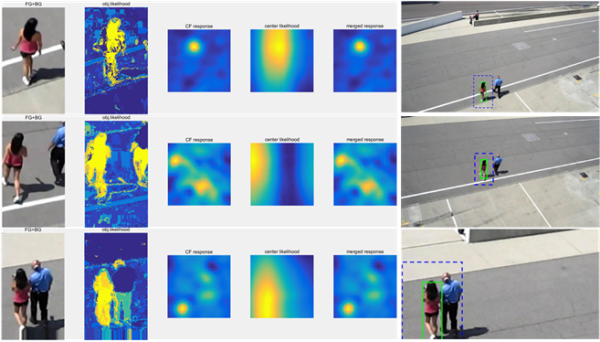
肯定是不一样的。1. CN特征是模板类特征，DAT中统计颜色直方图是概率特征，举个例子，跟踪目标是人，穿白色上衣，黑色裤子，CN特征学习到上白下黑的模板，颜色直方图学习到目标60%是白色，40%是黑色。如果这个人倒立了，CN模板对不上就认为这个不是目标，而颜色直方图依然60%白40%黑，判定就是目标。2. HOG+CN任然在相关滤波框架中，具有相关滤波的固有缺陷，而像素级概率颜色概率响应不受这一影响，能正确检测快速变形和快速运动情况，某种程度上来说DAT缓解了CF的边界效应，所以对快速运动情况不像DSST那么差。

**Staple为什么能控制计算量，比DSST和DAT都快还不损失性能？**

首先，Staple中检测和更新的图像块大小限制在150\*150以内，即HOG特征分辨率在38\*38以内，这样DSST的速度就会很快。但前面也分析过即使2因子下采样精度已经比较低了，那为什么Staple的精度不受影响呢？因为平移相关滤波响应和颜色概率响应要线性加权，而颜色概率响应得分辨率在150\*150左右下降不是很多，这就弥补了平移相关滤波的精度损失。举个例子，原始检测图像块分辨率是400\*400，如果限制分辨率到100\*100，HOG特征分辨率是25\*25，响应图也是25\*25，1像素位移对应跟踪框16像素位移；而加入DAT后，颜色概率响应分辨率是100\*100，相关滤波响应先上采样到100\*100，再两个响应结合，最终响应的分辨率是100\*100，1像素位移对应跟踪框4像素位移，跟踪精度提高了很多且保持低计算量。其次，像素级颜色概率图到颜色概率响应采用积分图加速计算，积分图是加速神器不用多说。这两种策略分别提高了DSST和DAT的速度，而且Staple并没有破坏封闭解，所以非常非常之NIUBILITY。

**两种响应图线性加权，会不会太简单，效果不够好？**

具体实现中采用比较简单的线性加权，最终响应 = (0.7\*相关滤波响应 + 0.3\*颜色概率响应)，可以看出Staple更加相信相关滤波的结果，而颜色概率只是起到辅助作用，看代码作者实现了另一种加权方式'fit\_gaussian'，可能效果不好最终没用。



相关滤波响应值是判别器输出的置信度，可以表示平移滤波器的可靠程度，跟踪越可靠峰值越大；颜色概率响应值表示当前像素点属于前景的置信度，像素点及附近像素属于前景的概率越高，响应值越大，也可以表示颜色概率模型的可靠程度。所以作者说这种线性加权是置信度加权，CF代表相关滤波的峰值点，PWP代表颜色概率响应的峰值点：

1、如果CF和PWP大小相似时，以相关滤波响应为主，颜色响应仅微调跟踪框的位置；

2、如果CF比较大，PWP比较小时，说明相关滤波响应更可靠，此时颜色概率响应几乎没有贡献；

3、如果CF比较小，PWP比较大时，说明颜色响应更可靠，以颜色峰值所在区域的CF局部峰值为主；

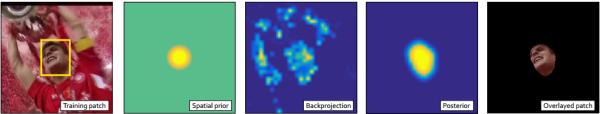
以上就是非常推荐而且比较快的Staple，在VOT数据库上性能非常高，竞赛结果已经看到了，前面也提到过颜色特征方法在OTB上不会很好，在OTB50上Staple略低于SRDCF。

Staple虽然简单高效，但也有不足之处，例如，仅以峰值判断置信度并不可靠，以上策略并不总是正确的。如果能有更全面可靠的指标，让我们知道相关滤波的检测置信度，那加权系数就可以自适应确定，在相关滤波检测置信度较低时更多地相信颜色概率的结果，也许会更加合理(个人拙见仅供参考)。也可以用更好的模板特征和统计特征在这个框架中完美结合，提升性能。

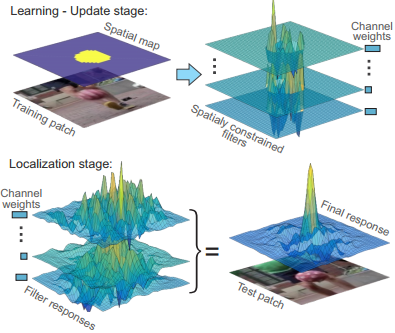
另一种相关滤波结合颜色概率的方法是CSR-DCF，基于DSST(当然也是没有Kernel)，提出了空域和通道可靠性

Lukežič A, Vojíř T, Čehovin L, et al. Discriminative Correlation Filter with Channel and Spatial Reliability [C]// CVPR, 2017.

空域可靠性通过前背景颜色直方图概率和中心先验计算空域二值约束掩膜，让滤波器自适应学习和跟踪颜色比较显著的目标部分，缓解边界效应：



通道可靠性用于区分检测时每个通道的权重，由两个指标决定：训练通道可靠性指标表示响应峰值越大的通道可靠性越高，检测可靠性指标表示响应图中第二和第一主模式之间的比值：



其实，CSR-DCF中的空域可靠性得到的二值掩膜就类似于CFLM中的掩膜矩阵P，在这里自适应选择更容易跟踪的目标区域且减小边界效应；以往多通道特征都是直接求和，而CSR-DCF中通道采用加权求和，而通道可靠性就是那个自适应加权系数。

CSR-DCF也属于伤筋动骨类方法，和CFLM一样也用ADMM迭代优化求解。同样是颜色方法，仅HOG+CN特征的CSR-DCF，在OTB100上接近SRDCF，在VOT2015上超过DeepSRDCF，在VOT2016上超过C-COT，速度13FPS，感觉也是非常NIUBILITY，等代码实测吧，如果非常注重性能，那这个接近实时的方法也非常推荐。

相关滤波还有个排在第一名的CCOT，下次在结合深度特征的相关滤波中一起介绍。