1. Introdution

绝大多数 MOT 都是遵循 TBD 方式，但是检测不总是精准的，这可能会实质上影响跟踪，效果，此外遮挡和形变也是 MOT 中面临的问题。

(a) (b)

Pair to consider	Switcher	Matching Confidence	
		With Switcher	Without Switcher
		0.7 High Confidence	0.2 Low Confidence

(c)

文中定义了两个不同的线索：

短期线索：是指相邻帧之间的更新线索。

长期线索：代表 tracklet-long 线索，其包含 tracklet 中目标的外观特征。

关于长短期线索：

SOTA 的单目标跟踪（**SOT**）方法可用于**捕获短期线索**，这对检测不精准时的情况很有帮助。如图 1(a)所示，检测器没有检测到目标，但 SOT 跟踪器可以找到目标，从而减少漏检。但是当发生遮挡时，大多数短期线索变得不可靠，这是因为包含遮挡区域会使 SOT 跟踪器漂移。而长期轨迹线索可以避免这种情况，如图(b)所示，长期线索仍保持稳定。

以前的工作没有充分利用这两个线索，作者通过实验发现，用于短期线索的 SOT 跟踪器无法区分相似的目标，而用于长期线索的网络无法越目标的精确位置，也就是说很难将所有线索有效结合到一个网络中。因此，作者提出了一个可以生成长短期线索，并自适应地选择它们进行数据关联地 MOT 统一框架。

关于 SAC(Switcher-Aware Classification):

除了长短期线索外，作者还利用了本地交互信息来解决 ID-switch 的问题。他们发现潜在的 switcher 对于正确匹配至关重要，如图 1(c)显示了 switcher 如何帮助匹配，作者使用 switcher-aware 分类器(通过增强决策树实现)来对潜在的 switcher 的信息进行编码，从而提升了跟踪器的鲁棒性。

本文有两个大的创新点

1) 利用 SOTA 的 SOT 捕获短期线索、ReID 捕获长期线索，并作出自适应决策以进行可靠跟踪。
2) 在数据关联阶段，提出的 SAC 会收集所有长短期线索，并考虑潜在的 switcher，最后生成分数来对潜在的 switcher 信息进行编码，从而提高了系统的鲁棒性。

2. Related Work

你不行，我很行。

3. The Proposed Framework

先进行问题公式化。一个被跟踪目标的轨迹可以被表示为$X = \{X_t\}$，其中$X_t = [X_t^x, X_t^y, X_t^w, X_t^h]$, t是帧索引, X_t^x, X_t^y是 bbox 左上角坐标, X_t^w, X_t^h是 bbox 的宽度和高度, qx是对目标X的整体跟踪质量， I_t^X是目标X在t帧的图像区域。

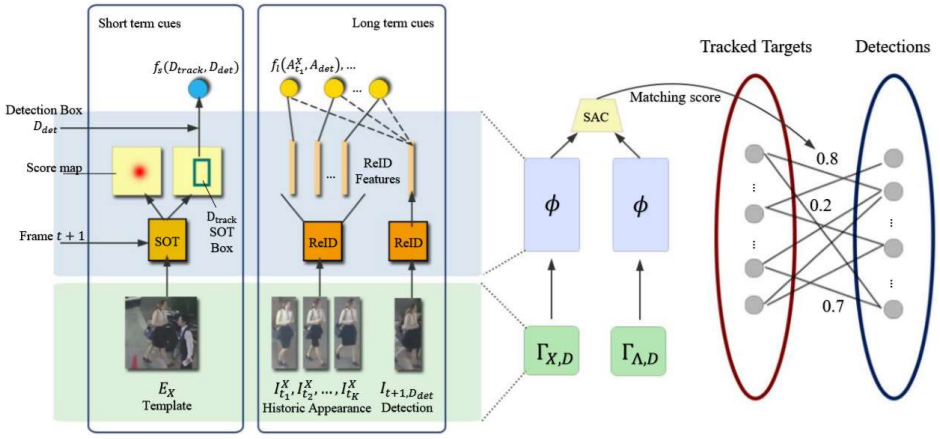


Figure 2. Overall design of the proposed MOT framework. Short term cues are captured by SOT sub-net and long term cues are captured by ReID sub-net. The switcher aware classifer (SAC) predicts whether the detection result matches the target using the short term and long term cues of the target, the detection result, and the switcher.

3.1 Overall Design

Step 1. 初始化，将被跟踪目标的集合 S 清空，并将 t=1

Step 2. 在 t 帧，对于一个目标 X，模板E_X是在下一帧I_{t+1}使用 SOT sub-net 搜寻的。SOT sub-net 输出帧I_{t+1}中模板最有可能的位置D_{track}。

Step 3. 对于在帧I_{t+1}中的检测结果D_{det}，其对应的检测图像区域$I_{t+1,D_{det}}$以及给目标的历史外观信息$\{I_{t_i}^X\}, i = 1,2,\dots,K$都会被 ReID sub-net 用来提取 long-term ReID features

Step 4. 目标的定位D_{track}是在 step2 中被 SOT 找到，而定位D_{det}是被检测器检测到的，step3 中获得的 ReID 特征将被组合到目标的匹配特征中去。

Step 5. 找到目标Λ的潜在 switcher，即最可能的 id 切换原因，并提取其 SOT 和 ReID 特征

Step 6. 借助 switcher 的匹配特征，目标的匹配特征由 SAC 使用，以生成关于检测结果是否与目标匹配的匹配分数。 对帧I_{t+1}中的每个检测结果重复此步骤，以获得其与跟踪目标的匹配分数。

Step 7. 使用在 Step 6 中获得的被跟踪目标与检测结果之间的匹配分数，来建立被跟踪目标与检测结果的二分图。使用 minimum-cost-network-flow 查找图的匹配结果。

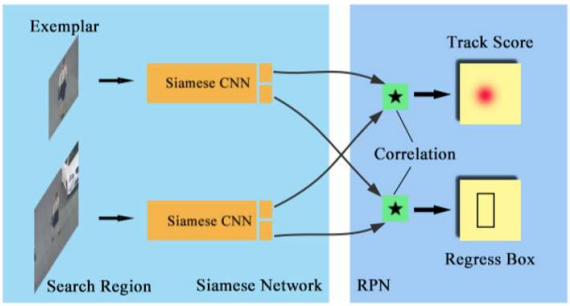
Step 8. 对于已经匹配的目标，使用匹配的检测信息来更新它的位置和模板。对于没有

匹配到的目标，使用 SOT 的结果来更新 tracklet，并丢弃不可靠或已丢失的目标。对于孤立的检测结果（新检测到的），如果其置信度得分满足新目标的条件，则将其添加到跟踪目标的集合中去。

Step 9.对下一帧重复 step2 到 step8，并将 t 更新 t=t+1,直到没有帧传入为止。

3.2 Using SOT Tracker for Short Term Cues

Baseline tracker。在我们的框架中，我们使用 Siamese-RPN 跟踪器来提取短期线索。按照原始结构，当前帧的模板E_X（也被称为示例）将尺寸调整为 127×127。要在下一帧搜索目标，搜索区域R需根据X的位置从帧I_{t+1}中裁剪。然后将搜索区域尺寸调整为 255×255。，具体来说，就是搜索区域的图片比例与模板（示例）相同。



(a)

Figure 3. Siamese-RPN architecture for SOT.

如图 3 所示，搜索区域和模板一同通过共享权重的 siamese CNN。然后，示例的 CNN 特征由两个分支（每个分支由两个卷积层组成）使用，与搜索区域相类似。一个分支用于获取得分图，另一个分支用于边界框回归。模板在搜索区域中的正确位置应该在得分图中具有最大得分。不同位置的边界框回归应指向此正确位置。

Short term features generation。SOT 子网输出 SOT 分数和预测的 bbox，称为 SOT box。要匹配的检测 bbox 称为 detection box。我们将 SOT 分数表示为p，将 SOT box 表示为D_{track}，将 detection box 表示为D_{det}，然后短期特征f_s的计算如下：

$$f_s(D_{track}, D_{det}) = IoU(D_{track}, D_{det}) \qquad (1)$$

Distractor-aware SOT tracker for MOT tracking。为了最大化 siamese RPN 的效果。我们修改了 anchor 的比例尺寸以适应行人目标。此外，我们使用行人数据完善网络。SOT 跟踪器的另一个问题是，**如果目标丢失了，很难确定何时停止跟踪**。当跟踪器漂移到背景干扰物的时候，跟踪器可能无法停止跟踪干扰物。为了使跟踪器分数分散注意力，我们**设计了跟踪分数修正策略**。我们使用在前一节中介绍的 step7 中找到的匹配结果来完善跟踪器得分。对于目标X，精确的总体跟踪质量q_X如下：

$$q_{X,t+1} = \begin{cases} \frac{q_{X,t} + IoU(D_{track}, D_{det}) \cdot p}{2}, \text{if matched,} \\ q_{X,t} \cdot decay \cdot p^k, \text{otherwise,} \end{cases} \qquad (2)$$

Decay 和 k 是用来处理不一致目标的超参数。这样，如果跟踪质量q_X低于阈值ζ_t，我们可以丢弃不可靠的目标。

3.3.Using ReID Network for Long Term Cues

我们使用 GoogLeNet Inception-v4 的修改版本作为 ReID 子网络的 backbone。ReID 特征是在分类之前的最后一层全连接层被提取出来的。Table 1 和 Table 2 展示了 backbone 的更多细节。

Type	Kernel/Stride	Output Size	Padding
input	- / -	$3 \times 224 \times 224$	-
convolution	7×7 / 2	$29 \times 112 \times 112$	3
pool	3×3 / 2	$29 \times 56 \times 56$	-
convolution	1×1 / 1	$27 \times 56 \times 56$	-
convolution	3×3 / 1	$142 \times 56 \times 56$	1
pool	3×3 / 2	$142 \times 28 \times 28$	-
inception-A	- / -	$379 \times 28 \times 28$	-
inception-A	- / -	$679 \times 28 \times 28$	-
pool	3×3 / 2	$679 \times 14 \times 14$	-
inception-A	- / -	$1037 \times 18 \times 18$	-
inception-A	- / -	$1002 \times 18 \times 18$	-
inception-A	- / -	$938 \times 18 \times 18$	-
inception-A	- / -	$861 \times 18 \times 18$	-
pool	14×14 / 1	$861 \times 1 \times 1$	-
fc	- / -	256	-

Table 1. The modified Inception-v4 architecture.

Step	Branch A	Branch B	Branch C	Branch D
1	conv 1×1	conv 3×3	conv 3×3	conv 1×1
2	-	conv 1×1	conv 3×3	pool 3×3
3	-	-	conv 1×1	-

Table 2. One inception-A block: all convolution layers of 3×3 and the pooling layer use padding 1, the others have no padding, and the pooling layer stride 2.

Quality-aware long term tracklet history construction. 要从目标的轨迹历史中选择K张图像，我们设计了一种质量感知机制。为了获得高质量的长期线索，我们使用质量过滤器在过去K个时间段内选择最佳的K张图像，以确保质量和鲁棒性。被选作目标的轨迹历史的K帧的索引$H = \{t_1 \ldots, t_K\}$表示。我们选择的K帧定义如下：

$$t_i = \underset{t-i\delta<\hat{t}\leq t-(i-1)\delta}{\arg\max} Q(I_{\hat{t}}^X), i = 1, 2, ..., K \qquad (3)$$

其中 Q 是输出质量得分的网络。Q 是使用 Resnet-18 模型实现的。I_t^X是目标X在t帧的图像区域。δ是决定选择间隔的超参数。例如，当$i = 1$，t_1是从$t, t-1, \ldots, t-\delta+1$中选取的最高质量分数帧。当$i = 2$，$t_2$是从$t-\delta, t-\delta-1, \ldots, t-2\delta+1$中选取的最高质量分数帧。因此，$t_i$中的$i$对应不同的步长和搜索范围。

Long-term feature generation。选择 K 张图像后，所有这些图像和要匹配的检测结果将被馈送到 ReID 子网并输出其 ReID 特征。 然后我们可以获得目标的 K 个长期特征，如下所示：

$$\begin{aligned} &\mathcal{F}_l^X = \{f_l(A_{t_i}^X A_{det}) | i = 1, \ldots, K\}, \\ &\text{where } f_l(A_{t_i}^X, A_{det}) = \frac{A_{t_i}^{X\mathrm{T}} \cdot A_{det}}{|A_{t_i}^X||A_{det}|}, \end{aligned} \qquad (4)$$

$A_{t_i}^X$是包含目标X的历史轨迹的第i张图片 ReID 特征的向量。A_{det}是检测结果的 ReID 特征。为了节省计算，每张轨迹中的图片只用 ReID 网络提取一次特征，这些的特征将会为将来的计算而保存下来。

3.4.Switcher-Aware Classifier

Switcher retrieval。我们观察了大量的身份转移(ID-Switch)，发现绝大多数的 ID-Switch 发生在两个目标相遇（即发生过大的遮挡重合）情况下。这启发了我们去标记与当前目标重合最大的其他目标，将其标记为最有可能潜在的 switcher。在数学上，对于每一个在t帧的轨迹X，它的位置被定义为X_t，潜在的 switcher 通过下式来获取：

$$\Lambda = \underset{Y \in \mathcal{S}\ s.t.\ Y \neq X}{\arg\max} IoU(X_t, Y_t). \qquad (5)$$

其中S是被跟踪目标的集合。

Input features。这里我们将两个子网络视为一个特征提取器ϕ，将目标X和检测结果D的两个子网络的输入表示为$\Gamma_{X,D}$，对于 switcher 相似操作。分类器的输入特征由两部分组成：主要考虑的目标的特征，表示为$\phi(\Gamma_{X,D})$，对于 switcher 的特征，定义为$\phi(\Gamma_{\Lambda,D})$。ϕ的定义如下：

$$\phi(\Gamma_{X,D}) = \{f_s(D_{track}, D_{det})\} \cup \mathcal{F}_l^X. \qquad (6)$$

$\phi(X,t)$的维数是$K+1$，对于 switcher 也是一样的。然后，我们通过将两部分结合起来以获得分类器的输入。

Classification。在分类步骤中，我们利用[8]提出的 regularized Newton boosting decision tree with weighted quantile sketch。如果分类结果y大于阈值ζ_m，则将成本为$1-y$的对应边添加到图中。

4.Experiments

4.1.ImplementDetails

该框架用 python 编写、Pytorch 支持。所有的 CNN 网络都先在 Imagenet 数据集上预训练，然后再在 MOT 任务中训练。此外，我们在训练两个子网络时使用了额外的私人数据，但是在训练分类器和质量滤波器时仅使用公共数据。我们使用的公共数据集是来自 MOT16 基准数据中的 7 个视频序列。而私有数据包含带标签的行人视频，与所有测试集视频完全不同。用于训练的私人数据量约为 100 分钟的 25FPS 视频，不同的行人数量约为 1000。

Tracklet status update。对于匹配的 tracklets，其位置将随着检测结果的位置而更新，并且 SOT 的模板也会更新。否则，位置保持不变，SOT 模板也保持不变。尽管结果匹配，但还是要使用卡尔曼滤波器对轨迹做平滑处理。质量衰减值为 0.95，指数$k=16$。当整体质量得分低于 0.5 时，这个目标将会被丢弃。匹配阈值$\zeta_m=0.05$，丢失阈值$\zeta_t=0.1$。

Training。我们使用与 SOT 子网络中的原始工作类似的方法，但是将 anchor 比例改为[1.0,1.6,2.2,2.8,3.4]训练 ReID 子网络时，我们使用随机梯度下降(SGD)优化器，并将初始学习率设置为 0.1，每 8 个 epoch 衰减 0.5 倍，总共 96 个 epoch，ReID 子网络的训练权重衰减为5×10^{-4}，mini-batch size 为 32。

我们在 xgboot 框架下训练了 SAC。树的数量设置为 410。最大深度为 5，学习率是 0.05，最小叶子结点权重设置为 1。

对于长期线索，可以将 tracklet 历史的选择间隔设置为 10 到 20.我们仅简单采样$K=3$帧，并设置$\delta=15$。

4.2.Evaluation on MOT Benchmarks

Datasets。我们提出的框架在 MOT16 和 MOT17 基准测试集上评估。它们共享相同的测试视频但是提供不同的检测输入。MOT17 修复了 ground truth，并使其更加准确。这些测试视频包括各种复杂的场景，这些仍是一个巨大的挑战。

Evaluate Metrics。按照基准测试，我们使用 CLEAR MOT 指标来评估我们的工作。在这

些指标中，MOTA 和 IDF1 被认为是最重要的。MOTA 包含了召回率，精准度以及 ID-Switch。IDF1 表示平均最大一致性跟踪率。在对 MOT 跟踪器评估时，标准 MOTA 与检测的召回率、精准度高度相关，而 ID-Switch 与其则相关度很低。然而，IDF1 可以证明一致性。强大的跟踪系统应同时具有较高的 MOTA 和 IDF1 分数。

Results。表 3 和表 4 展示了在线和批处理方式分别在 MOT16 和 MOT17 上的表现。此外，在带有专用探测器的 MOT16 任务中，我们的跟踪器以及 KDNT，LMPp，HT SJTUZTE 和 POI 跟踪器使用的是与 POI 跟踪器作者建议相同的检测器。 检测结果可在 Google Drive 得到。

Benchmark	Method	**MOTA** ↑	MOTP ↑	**IDF1** ↑	IDP ↑	IDR ↑	FP ↓	FN ↓	IDS ↓
MOT16	RAR16pub[12]	45.9%	74.8%	48.8%	69.7%	37.5%	6871	91173	648
MOT16	STAM16[10]	46.0%	74.9%	50.0%	71.5%	38.5%	6895	91117	473
MOT16	DMMOT[42]	46.1%	73.8%	54.8%	77.2%	42.5%	7909	89874	532
MOT16	AMIR[30]	47.2%	**75.8%**	46.3%	68.9%	34.8%	**2681**	92856	774
MOT16	MOTDT[24]	47.6%	74.8%	50.9%	69.2%	40.3%	9253	85431	792
MOT16	Ours	44.8%	75.1%	53.8%	75.2%	41.8%	9639	90571	**451**
MOT16	Ours(with filter)	**49.2%**	74.0%	**56.5%**	**77.5%**	**44.5%**	7187	**84875**	606
MOT17	PHD_GSDL17[13]	48.0%	**77.2%**	49.6%	68.4%	39.0%	23199	265954	3998
MOT17	AM_ADM17[21]	48.1%	76.7%	52.1%	71.4%	41.0%	25061	265495	2214
MOT17	DMAN[42]	48.2%	75.7%	55.7%	75.9%	44.0%	26218	263608	2194
MOT17	HAM_SADF17[39]	48.3%	**77.2%**	51.1%	71.2%	39.9%	**20967**	269038	1871
MOT17	MOTDT17[24]	50.9%	76.6%	52.7%	70.4%	42.1%	24069	250768	2474
MOT17	Ours	50.3%	76.8%	56.3%	**76.5%**	44.6%	21345	257062	**1815**
MOT17	Ours(with filter)	**52.7%**	76.2%	**57.9%**	76.3%	**46.6%**	22512	**241936**	2167
MOT16p	EAMTT_16[31]	52.5%	78.8%	53.3%	72.7%	42.1%	**4407**	81223	910
MOT16p	SORTwHPD16[5]	59.8%	**79.6%**	53.8%	65.2%	45.7%	8698	63245	1423
MOT16p	DeepSORT_2[36]	61.4%	79.1%	62.2%	72.1%	54.7%	12852	56668	781
MOT16p	RAR16wVGG[12]	63.0%	78.8%	63.8%	72.6%	56.9%	13663	53248	**482**
MOT16p	CNNMTT[25]	65.2%	78.4%	62.2%	73.7%	53.8%	6578	55896	946
MOT16p	POI[40]	66.1%	79.5%	65.1%	**77.7%**	56.0%	5061	55914	805
MOT16p	Ours	**69.6%**	78.5%	**68.6%**	77.1%	**61.7%**	9138	**45497**	768

Table 3. Comparision between the proposed MOT framework (online mode) with other online processing SOTA methods in MOT16 and MOT17. 'with filter' means detection score refiner is used. 'MOT16p' means MOT16 with private detection. Red for the best result.

Benchmark	Method	**MOTA** ↑	MOTP ↑	**IDF1** ↑	IDP ↑	IDR ↑	FP ↓	FN ↓	IDS ↓
MOT17	IOU17[6]	45.5%	76.9%	39.4%	56.4%	30.3%	**19993**	281643	5988
MOT17	MHT_bLSTM[19]	47.5%	**77.5%**	51.9%	71.4%	40.8%	25981	268042	2069
MOT17	EDMT17[7]	50.0%	77.3%	51.3%	67.0%	41.5%	32279	247297	2264
MOT17	MHT_DAM_17[18]	50.7%	**77.5%**	47.2%	63.4%	37.6%	22875	252889	2314
MOT17	jCC[17]	51.2%	75.9%	54.5%	72.2%	43.8%	25937	247822	1802
MOT17	FWT_17[15]	51.3%	77.0%	47.6%	63.2%	38.1%	24101	247921	2648
MOT17	Ours(with filter)	**54.7%**	75.9%	**62.3%**	**79.7%**	**51.1%**	26091	**228434**	**1243**
MOT16p	NOMTwSDP16[9]	62.2%	79.6%	62.6%	77.2%	52.6%	**5119**	63352	406
MOT16p	MCMOT_HDM[20]	62.4%	78.3%	51.6%	60.7%	44.9%	9855	57257	1394
MOT16p	KDNT[40]	68.2%	79.4%	60.0%	66.9%	54.4%	11479	45605	933
MOT16p	LMP_p[35]	71.0%	**80.2%**	70.1%	78.9%	63.0%	7880	44564	**434**
MOT16p	HT_SJTUZTE[23]	**71.3%**	79.3%	67.6%	75.2%	61.4%	9238	42521	617
MOT16p	Ours	71.2%	78.3%	**73.1%**	**80.7%**	**66.8%**	10274	**41732**	510

Table 4. Comparision between the proposed MOT framework (batch mode) with other batch processing SOTA methods in MOT16 and MOT17. 'with filter' means detection score refiner is used. 'MOT16p' means MOT16 with private detection. Red for the best result.

结果表明，在 MOT16 和 MOT17 基准测试中，我们的框架均优于许多以前的 STOA 跟踪器。 在在线/批量处理算法中，MOTA 和 IDF1 分数在 MOT16 / MOT17 中均处于领先地位。

4.3.Ablation Study and Discussion

How do different cues ifluence the tracking quality?

在 MOT16 训练集上评估了 ablation 研究。 由于我们已使用训练集进行验证，因此在训练图 4 中所有 ablation study 结果的子网，可感知切换器的分类器和质量过滤器时，我们将 MOT16 训练集从训练数据中排除掉。对于分类器和质量过滤器， 我们在 ablation study 中使用了额外的私人训练数据。

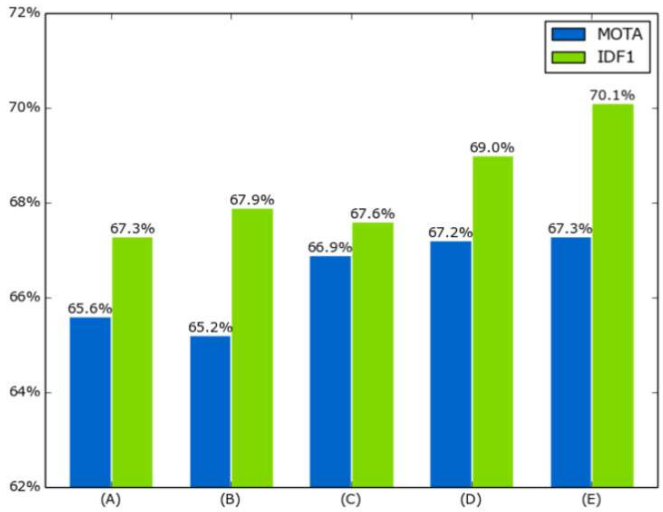


Figure 4. Analysis of our framework using different components. (A) baseline (hand-crafted) (B) long-term cues only (C) short-term cues only (D) long and short term cues (E) long and short term cues with SAC

图 4 显示了不同组件的影响。基线模型（图 4 中的（A））没有使用学习的分类器，而是仅使用位置信息以手工方式计算 affinity。图 4 中的其他实验结果共享相同的框架设置，不同之处在于分类器的输入特征，并且训练了四个不同的分类器以适应不同的输入特征。从图可以看出，与基线相比，短期线索对 MOTA 的改善要比长期线索更大。这也是两个相邻检测框之间相关性的最直观反映。当有效利用短期线索时，MOTA 分数提高了 1.3%，IDF1 提高了 0.3%。另一方面，结合长期线索可以有效地提高 tracklets 之间的判别能力，从而使 IDF1 增加 1.4%。但是，从长期线索获得的 MOTA 改进要小于从短期线索获得的 MOTA 改进。将短期和长期线索结合起来要比使用单个线索更好，这证明了这两个线索是相互补充的。第三，添加支持切换器的分类器可以大大减少 ID 切换器的数量，从而导致 IDF1 又增加 1.1%，而对 MOTA 的影响很小。结合长期和短期线索的学习方法是有效的，并且使用 SAC 的结合带来了对多个指标的改进。

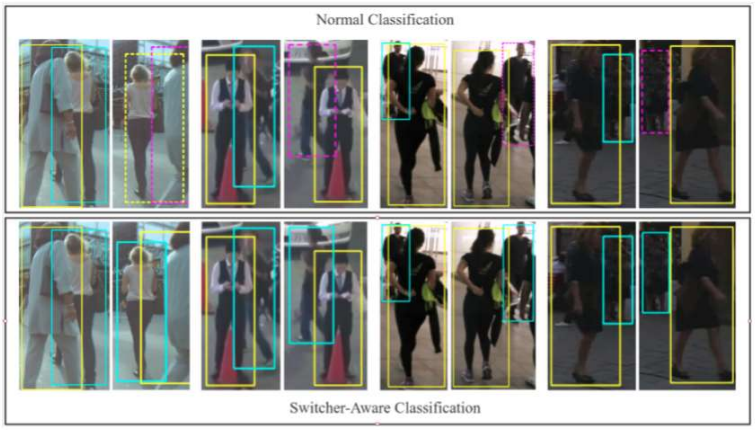


Figure 5. Examples of identity switch cases (top row) fixed when SAC is used (bottom row) for matching targets between adjacent frames. Dashed line boxes indicate id-switch.

How does SAC work?

我们还将分析视频的真实效果。 图 5 显示，通过具有切换器感知功能的分类，跟踪更加可靠。 SAC 的主要贡献在于它固定了许多 ID-switches。 使用 SAC 后，在私有 MOT16 中，ID 交换号从 642 减少到 569，IDF1 增加 1.1％。这是因为在传统的成对匹配中，与本地 switcher 之间缺乏比较，会在 pair 被遮挡并因此判断为不匹配时带来错误。 此外，当发生遮挡时，SAC 有助于区分不同的目标。

Can multiple cues been handle in one network?

我们试图从 SOT 主干 CNN 中提取一些功能，并通过 ROI 池层与 ReID 分支结合。 实验表明，对 ReID 和 SOT 任务进行多任务训练会导致 SOT 和 ReID 准确性均下降。 用 MOT 任务中的多任务训练网络代替 Siamese-RPN，MOTA 减少了 0.6％，IDF1 减少了 2.5％。 SOT 任务需要背景知识，而 ReID 任务则需要消除背景知识，如果两个任务都使用单个网络，则在特征学习期间会引起冲突。 暂时很难在一个网络中处理两个线索，因此需要做进一步的研究工作。

Why do we need small feature dimension?

我们使用小维度的输入特征来平衡不同信息的特征长度。众所周知，运动和位置特征通常非常短。如果我们将它们与长的外观特征结合在一起，则很难充分利用位置和运动特征。我们尝试将原始 ReID 特征与短的位置特征直接连接，实验结果表明，不平衡的输入特征使 IDF1 降低 1.3％，MOTA 降低 0.1％，IDS 数量增加 88。我们认为位置和运动信息非常重要，应该通过减小外观特征的尺寸来强调它们。另一个不可避免的问题是数据关联的复杂性。简短的特征使该过程更快。

Why we use boosting trees for classfication?

我们尝试了其他不同的分类器，例如线性层的神经网络（NN），支持向量机（SVM），但增强决策树（BDT）是最好的分类器。 使用包括 MOT16 在内的完整训练数据完成了该实验。 NN，SVM 和 BDT 的 MOTA 分别为 67.0％，67.6％和 67.8％。 我们已经发现，对于这么小的输入特征，神经网络在如此小的规模数据集上的表现并不理想。

5.Conclusions

在本文中，我们提出了一个有效的 MOT 框架，该框架学习了长期和短期线索的结合。长期提示可以帮助纠正短期提示的错误，例如，避免 SOT 子网在遮挡期间漂移。 我们还提出了一种切换器感知的分类方法，以提高跟踪系统的鲁棒性。 MOT 基准的出色表现证明了所提出框架的有效性。