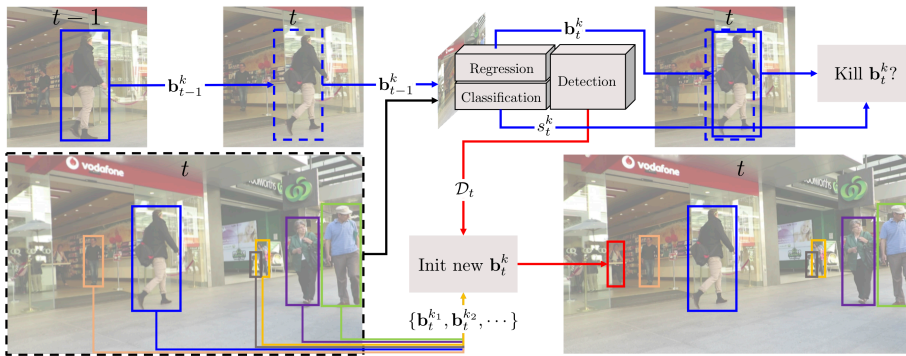


---

## Tracking without bells and whistles

This repository provides the implementation of our paper **Tracking without bells and whistles** (Philipp Bergmann, Tim Meinhardt, Laura Leal-Taixe) [<https://arxiv.org/abs/1903.05625>]. This branch includes an updated version of Tracktor for PyTorch 1.3 with an improved object detector. The original results of the paper were produced with the `iccv_19` branch.

In addition to our supplementary document, we provide an illustrative web-video-collection. The collection includes exemplary Tracktor++ tracking results and multiple video examples to accompany our analysis of state-of-the-art tracking methods.



The presented Tracktor accomplishes multi-object tracking only with an object detector and consists of two primary processing steps, indicated in blue and red, for a given frame  $t$ . First, the regression of the object detector aligns already existing track bounding boxes  $\mathbf{b}_{t-1}^k$  of frame  $t-1$  to the object's new position at frame  $t$ . The corresponding object classification scores  $s_t^k$  of the new bounding box positions are then used to kill potentially occluded tracks. Second, the object detector (or a given set of public detections) provides a set of detections  $\mathcal{D}_t$  of frame  $t$ . Finally, a new track is initialized if a detection has no substantial Intersection over Union with any bounding box of the set of active tracks  $B_t = \{\mathbf{b}_t^{k1}, \mathbf{b}_t^{k2}, \dots\}$ .

## Installation

1. Clone and enter this repository: `git clone https://github.com/phil-bermann/tracking_wo_bnw` `cd tracking_wo_bnw`
2. Install packages for Python 3.7 in virtualenv:
  1. `pip3 install -r requirements.txt`
  2. Install PyTorch 1.7 and torchvision 0.8 from [here](#).
  3. Install Tracktor: `pip3 install -e .`
3. MOTChallenge data:
  1. Download 2DMOT2015, MOT16, MOT17Det, MOT17, MOT20Det and MOT20 and place them in the `data` folder.
  2. Unzip all the data in the `data` directory.

- 
4. Download model (MOT17 object detector, MOT20 object detector, and re-identification network) and MOTChallenge result files:

1. Download zip file from [here](#).
2. Extract in `output` directory.

## Evaluate Tracktor

In order to configure, organize, log and reproduce our computational experiments, we structured our code with the Sacred framework. For a detailed explanation of the Sacred interface please read its documentation.

1. Tracktor can be configured by changing the corresponding `experiments/cfgs/tracktor.yaml` config file. The default configuration runs Tracktor++ with the FPN object detector as described in the paper.
2. The default configuration is `Tracktor++`. Run `Tracktor++` by executing:

```
1 python experiments/scripts/test_tracktor.py
```

3. The results are logged in the corresponding `output` directory.

For reproducibility, we provide the new result metrics of this updated code base on the `MOT17` challenge. It should be noted, that these surpass the original Tracktor results. This is due to the newly trained object detector. This version of Tracktor does not differ conceptually from the original ICCV 2019 version (see branch `iccv_19`). The results on the official MOTChallenge webpage are denoted as the `Tracktor++v2` tracker. The train and test results are:

```
1 ***** MOT17 TRAIN Results *****
2 IDF1  IDP  IDR| RcLl  Prcn  GT  MT  PT  ML|  FP  FN  IDs  FM
3 | MOTA  MOTP  MOTAL
3 65.2 83.8 53.3| 63.1 99.2 1638 550 714 374| 1732 124291 903
4 | 1258| 62.3 89.6 62.6
4
5 ***** MOT17 TEST Results *****
6 IDF1  IDP  IDR| RcLl  Prcn  GT  MT  PT  ML|  FP  FN  IDs  FM
7 | MOTA  MOTP  MOTAL
7 55.1 73.6 44.1| 58.3 97.4 2355 498 1026 831| 8866 235449 1987
8 | 3763| 56.3 78.8 56.7
```

We complement the results presented in the paper with `MOT20` train and test sequence results. To this end, we run the same tracking pipeline as for `MOT17` but apply an object detector model trained on the `MOT20` training sequences. The corresponding model file is the same as used for this work.

```
1 ***** MOT20 TRAIN Results *****
```

---

2	IDF1	IDP	IDR	Rc1l	Prcn	GT	MT	PT	ML	FP	FN	IDs	FM
	MOTA												
3	60.7	73.4	51.7	68.5	97.4	2212	892	1064	259	20860	357227	2664	6504
	66.4												
4													
5	***** MOT20 TEST Results *****												
6	IDF1	IDP	IDR	Rc1l	Prcn	GT	MT	PT	ML	FP	FN	IDs	FM
	MOTA												
7	52.6	73.7	41.0	54.3	97.6	1242	365	546	331	6930	236680	1648	4374
	52.6												

## Train and test object detector (Faster R-CNN with FPN)

For the object detector, we followed the new native `torchvision` implementations of Faster R-CNN with FPN which are pre-trained on COCO. The provided object detection model was trained and tested with the `experiments/scripts/faster_rcnn_fpn_training.ipynb` Jupyter notebook. The object detection results on the MOT17Det train and test sets are:

1	***** MOT17Det TRAIN Results *****									
2	Average Precision: 0.9090									
3	Rc1l	Prcn	FAR	GT	TP	FP	FN	MODA	MODP	
4	97.9	93.8	0.81	66393	64989	4330	1404	91.4	87.4	
5										
6	***** MOT17Det TEST Results *****									
7	Average Precision: 0.8150									
8	Rc1l	Prcn	FAR	GT	TP	FP	FN	MODA	MODP	
9	86.5	88.3	2.23	114564	99132	13184	15432	75.0	78.3	

## Training the re-identification model

1. The training config file is located at `experiments/cfgs/reid.yaml`.
2. Create reID data `python experiments/evaluation_tools/reid_mot_to_coco_gt.py --dataset MOT17 --data_root data`
3. Start training by executing `python experiments/scripts/train_reid.py`.

## Publication

If you use this software in your research, please cite our publication:

1	@InProceedings{tracktor_2019_ICCV,
2	author = {Bergmann, Philipp and Meinhardt, Tim and Leal{-}Taix{'e }}, Laura},

---

```
3  title = {Tracking without bells and whistles},
4  booktitle = {The IEEE International Conference on Computer Vision (
    ICCV)},
5  month = {October},
6  year = {2019}}
```