Master in Computer Vision Barcelona

Week 5: Multi-target multi-camera tracking

Team 3

Gemma Alaix Granell: gemma.alaix@e-campus.uab.cat
Josep Brugués i Pujolràs: josep.brugues@e-campus.uab.cat
Clara Garcia Moll: clara.garciamo@e-campus.uab.cat
Aitor Sánchez Abellán: aitor.sancheza@e-campus.uab.cat

Contents

1. **T1: Multi-Target Single-Camera (MTSC)**
   a. Tracking
   b. Object Detection
   c. Bounding Box Filtering
   d. Tracking Enhancement
2. **T2: Multi-Target Multi-Camera (MTMC) Tracking**
   a. Matching
   b. Clustering
   c. ReID + GPS
3. **Conclusions**

T1: Multi-Target Single-Camera (MTSC) Tracking

1. **Object Detection**: Two models, **YoloV3** and **EfficientDet**, have been re-trained to detect the cars.
2. **Bounding Box Filtering**: By using **ROI** information, cars outside the road are filtered.
3. **Tracking**: Overlapping and Kalman techniques has been tested. Moreover, OF results from MaskFlowNet has been used to improve the results.
4. **Enhancement**: To further improve the results, several techniques are proposed.

Car detection → **Bounding Box filtering** → **Tracking** → **Enhancement**

- YoloV3 (Ultralytics)
- EfficientDet (TensorFlow)

- Parked based on ROI

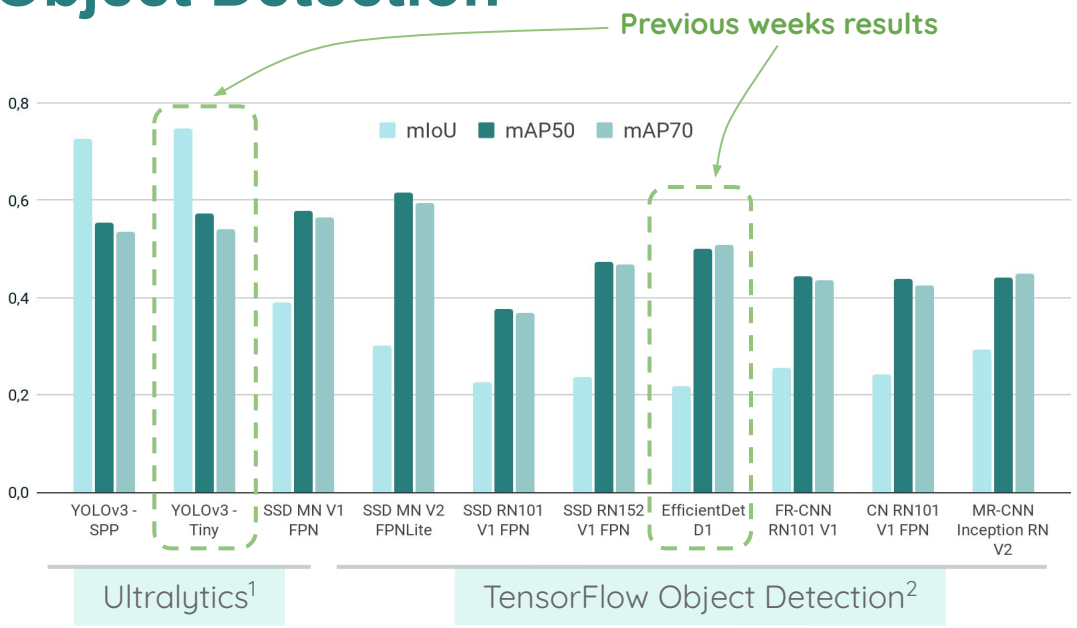
- Overlapping (ours)
- Kalman (Sort)
- Overlapping + MaskFlowNet (OF)

- Interpolation
- Noise removal
- Parked based on trajectory

Aitor Sánchez

T1.1: Object Detection

[1] https://github.com/ultralytics/yolov3
[2] https://github.com/tensorflow/models



Object Detectors
YOLOv3
SSD
EfficientDet
Faster R-CNN
CenterNet
Mask R-CNN

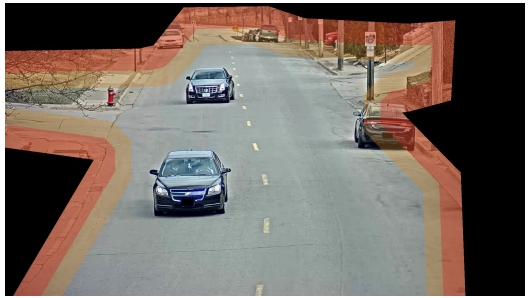
- Best method previously tested from **Ultralytics** and **TensorFlow** has been selected.
- **Re-trained for each sequence:** e.g. For **S03, S01** and **S04** are used for training (**20/80** validation split for each camera). Repeated for **S01** and **S04.**

T1.2: Bounding Box Filtering

1. To prevent detection data to include the **parked cars** (not included on the GT) a filter after loading the detections is applied.
2. For each frame, a **distance matrix** is computed. It calculates the **Euclidian distance** from each pixel to the closest ROI border.
3. Then, for each bounding box:
   a. If **roi_distance from detection centroid px < threshold**: detection = **Parked**

Distance to ROI border (px)

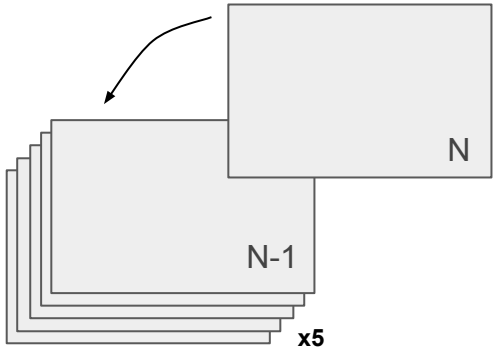
Thresholds Example

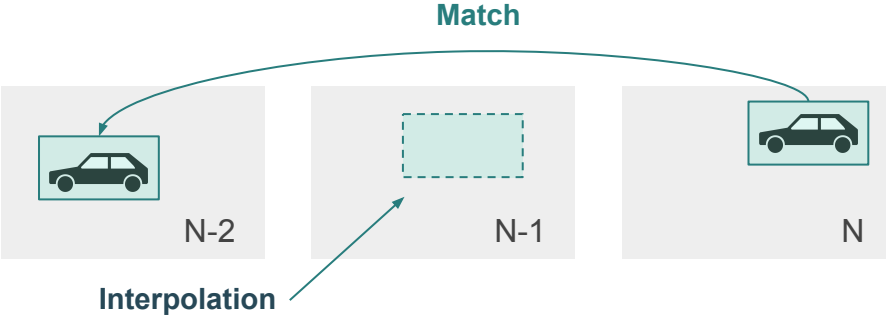


T1.3: Tracking by Maximum Overlap

Improved Overlap algorithm:

1. **Initialization**: Assign a unique ID to each detected object at the first frame.
2. Assign, at frame N, the same ID to the detection with the highest overlap (IoU) in frame N-1. **It Only considers IoUs > 0.5**.
3. If no match, look backwards **up to N-5** for a match with IoU>0.5.
   a. If match, assign the corresponding ID and **interpolate detections between the intermediated frames**.
   b. Else, assign a **new ID**.
4. Return to 2.





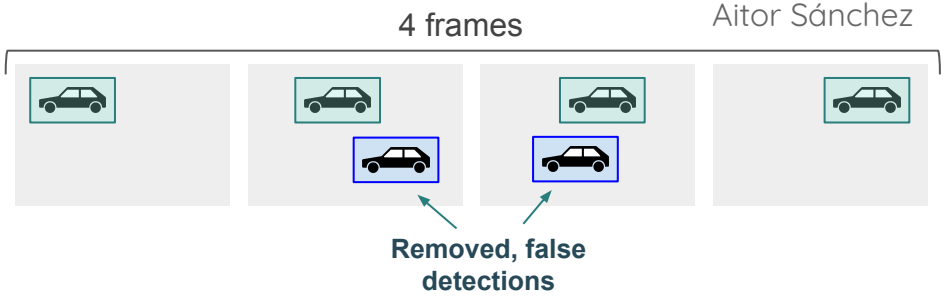
T1.4: Enhancements

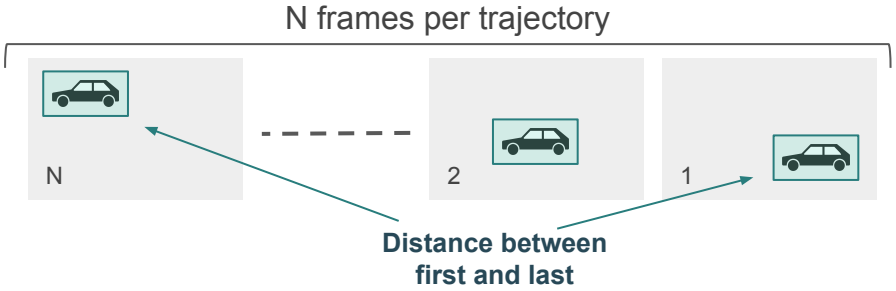
Aitor Sánchez

Enhancements

Noise filtering:

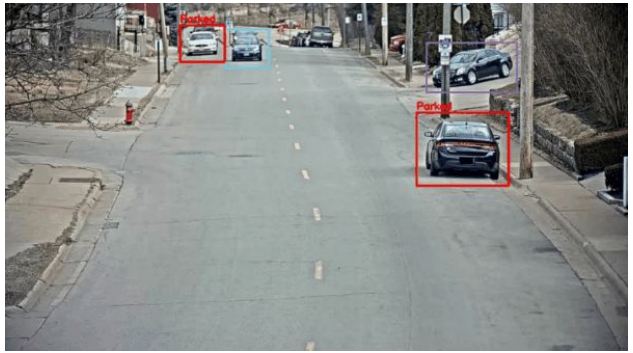
1. Keep track of the ID occurrences along the sequence.
2. Get rid of detections which appear in **less than 4 frames** (after interpolation).





Parked cars by displacement.

1. For each trajectory (ID assigned) get the location of the first and the last bbox.
2. Compute the **distance** (squared distance)
   A. If Distance > threshold then ID considered **parked car**.



T1: Multi-Target Single-Camera (MTSC) Tracking

Model	Technique	IDF1 S01					
		c001	c002	c003	c004	c005	Avg
YoloV3	Overlap	0.4595	0.5708	0.3839	0.4746	0.2596	**0.4297**
	Kalman	0.5952	0.6008	0.4763	0.5167	0.3546	**0.5087**
	OF-Overlap	0.4017	0.5790	0.3842	0.4745	0.2587	**0.4196**
EfficientDet	Overlap	0.3740	0.4840	0.3559	0.3996	0.2085	**0.3644**
	Kalman	0.5053	0.4939	0.4029	0.4552	0.2904	**0.4295**
	OF-Overlap	0.3242	0.5136	0.3135	0.4043	0.2302	**0.3572**

- The best result is obtained by **Kalman**, where the IDF1 is slightly higher for all cameras.
- Moreover, the average obtained using Overlap and Overlap + OF are very similar.
- Despite the fact that YoloV3 obtains better results, the IDF1 for both models follow a similar pattern.

T1: Multi-Target Single-Camera (MTSC) Tracking

Model	Technique	IDF1 S03						
		c010	c011	c012	c013	c014	c015	Avg
YoloV3	Overlap	0.9391	0.7990	0.3185	0.7742	0.5632	1.0	**0.7323**
	Kalman	0.8236	0.7347	0.3536	0.7708	0.6988	1.0	**0.7303**
	OF-Overlap	0.940	0.7985	0.3320	0.7743	0.5632	1.0	**0.7347**
EfficientDet	Overlap	0.7738	0.6855	0.2723	0.6403	0.4849	1.0	**0.6428**
	Kalman	0.7009	0.6127	0.2945	0.6390	0.6390	1.0	**0.6477**
	OF-Overlap	0.784	0.6955	0.2822	0.6922	0.4917	1.0	**0.6576**

- The best result is obtained by **Overlap + OF**, where the average of the IDF1 metric is higher.
- Moreover, almost all results **go hand by hand**. Except in camera C014 where the Kalman obtain slightly higher result.
- Furthermore, as it is seen, the results obtained using the detections of **YoloV3 are better than the ones obtained using detections of EfficientDet**. Which is related to the mAP obtained in each case.

T1: Multi-Target Single-Camera (MTSC) Tracking

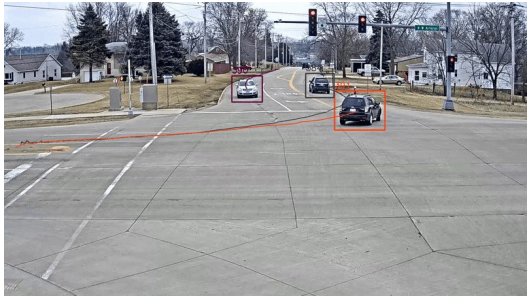
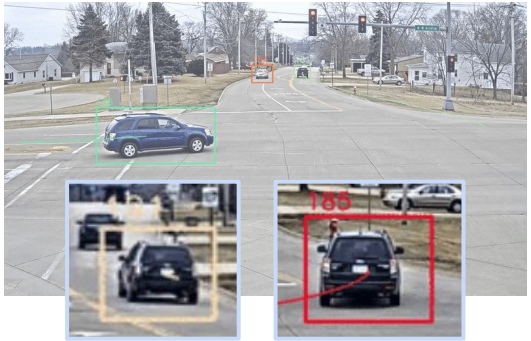
	IDF1 S04												
Technique	**c016**	**c017**	**c018**	**c019**	**c020**	**c021**	**c022**	**c023**	**c024**	**c025**	**c026**	**c027**	**c028**
Overlap	0.568	0.681	0.603	0.940	0.583	0.364	0.707	0.932	0.774	0.685	0.574	0.407	0.959
Kalman	0.446	0.686	0.602	0.926	0.429	0.365	0.551	0.815	0.840	0.611	0.744	0.455	0.962
OF-Overlap	0.568	0.681	0.603	0.940	0.583	0.364	0.707	0.933	0.733	0.685	0.573	0.407	0.958

Technique	**c029**	**c030**	**c031**	**c032**	**c033**	**c034**	**c035**	**c036**	**c037**	**c038**	**c039**	**c040**	**Avg**
Overlap	0.262	0.447	0.818	0.612	0.155	0.202	0.279	0.211	0.346	0.414	0.784	0.679	**0.559**
Kalman	0.316	0.509	0.554	0.534	0.192	0.220	0.361	0.203	0.330	0.443	0.791	0.638	**0.541**
OF-Overlap	0.262	0.447	0.818	0.612	0.155	0.202	0.280	0.211	0.357	0.414	0.782	0.679	**0.558**

- The best result is obtained by **Overlap**, where the average of the IDF1 is slightly higher.
- Moreover, the average obtained using Overlap and Overlap + OF **are very similar**.
- Furthermore, the results of each camera using the different methods **follow the same pattern**.

T1: Multi-Target Single-Camera (MTSC) Tracking

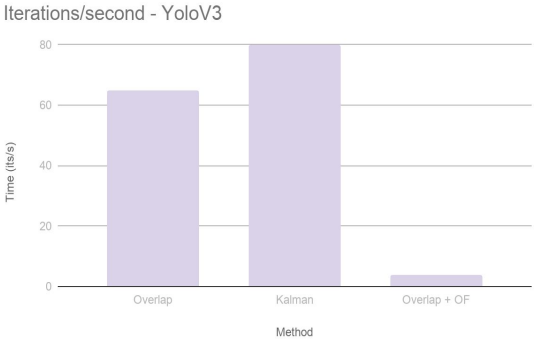
Qualitative Results

	S01	S03	S04
Overlap			
Kalman			

- When the object follows a **linear motion**, the car tracking is **estimated accurately**.
- If the motion is **not linear**, the track could be **lost and the id changes**.

T1: Multi-Target Single-Camera (MTSC) Tracking

- The results obtained in S03 are better because almost all detection **motions are linear**.
- The results for the different methods in each Sequence are really **similar and follow a same pattern**.
- Computational cost is much **more expensive** using Overlap + OF.
- **YoloV3 works better than EfficientDet**, which is related with the mAP obtained after training with both methods.



Gemma Alaix

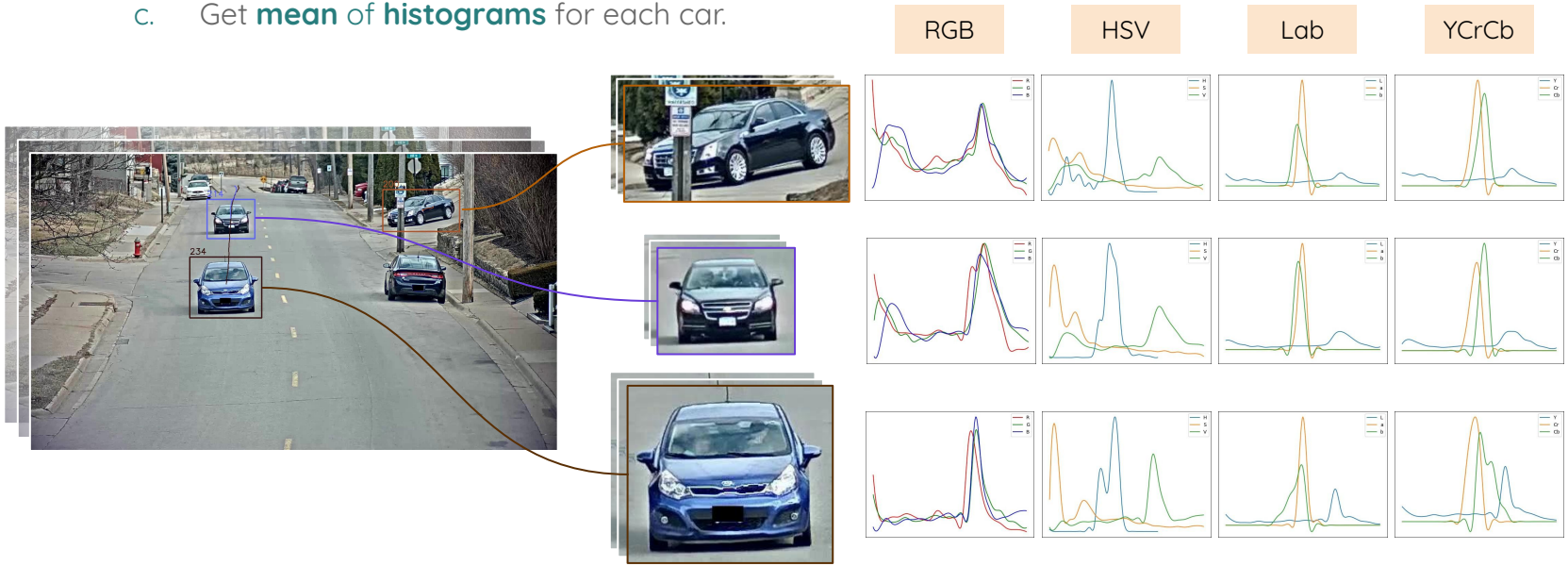
T2: Multi-Target Multi-Camera (MTMC) Tracking

Color histogram-based

1. Multi-Target **Single-Camera** (MTSC) Tracking.
2. For each camera:
   a. **Join car crops** based on ids.
   b. Compute **histogram** for each **crop**.
   c. Get **mean** of **histograms** for each car.

Color Spaces (models)

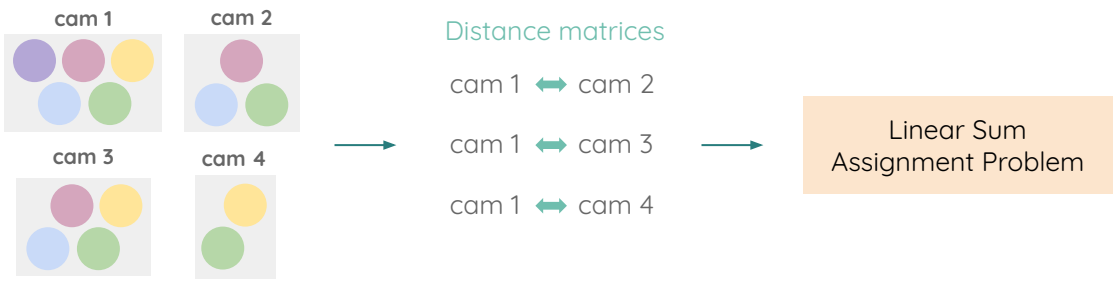
Chromaticity Primaries Excluding luminance **R**ed **G**reen **B**lue	Based on human perception Perceptually linear **L**uminance, **a** and **b** "color"
Color in terms of Hue Saturation +- color **H**ue **S**aturation **V**alue	Separate lightness from chroma in an RGB input **Y** luma, **Cr** and **Cb** chrominance



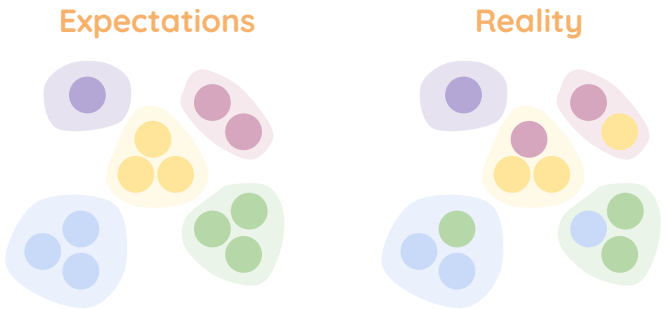
T2: Multi-Target Multi-Camera (MTMC) Tracking

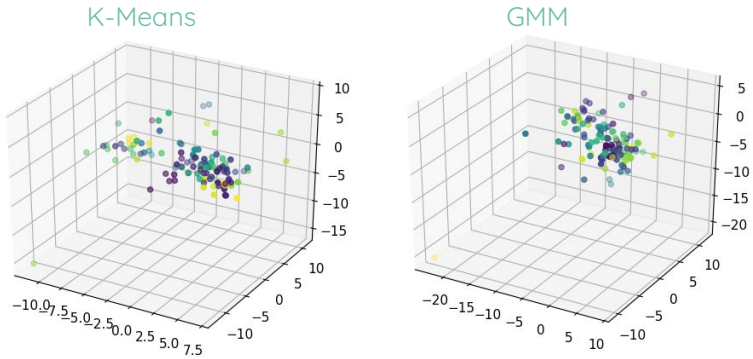
3. **Match** car sequences within cameras:

   a. **Distance matrix** of histograms (Euclidean distance, L1 distance, Bray–Curtis)



   b. **Clustering** of histograms (K-Means, Gaussian Mixture Models).





Plots from tracking with kalman and using Lab color space

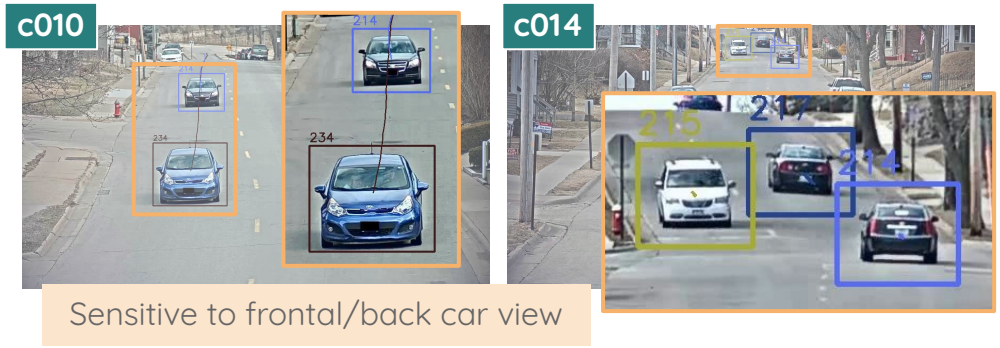
T2: Multi-Target Multi-Camera (MTMC) Tracking

Main Drawbacks

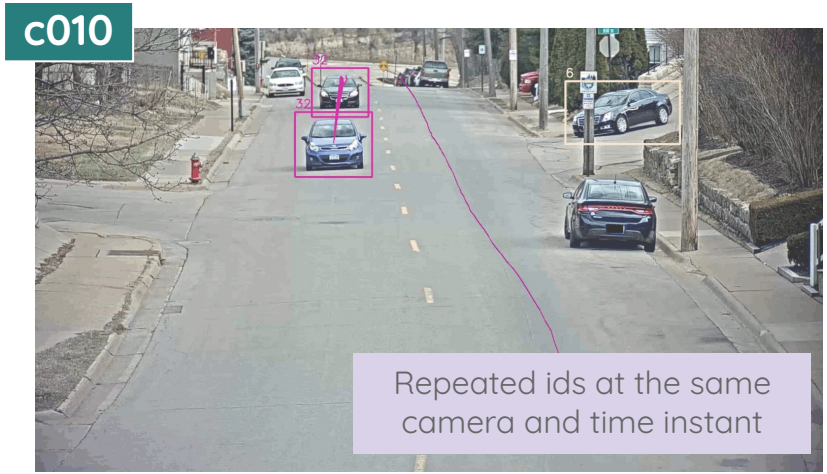
- Area cropped includes too much background.
- Color histograms are poor descriptors for such amount of different cars.
- Mean of histograms does not generalize well car features (3D objects).
- Camera with largest number of cars does not necessarily include all the cars in the scene.
- Temporal information not taken into account.

Improvements

- Segmentation on cropped areas.
- Use other descriptors, e.g. texture or CNN.
- Cluster with information from all frames.
- Include time constraints:
  - Restrict car apparison in the cam just once.



Sensitive to frontal/back car view



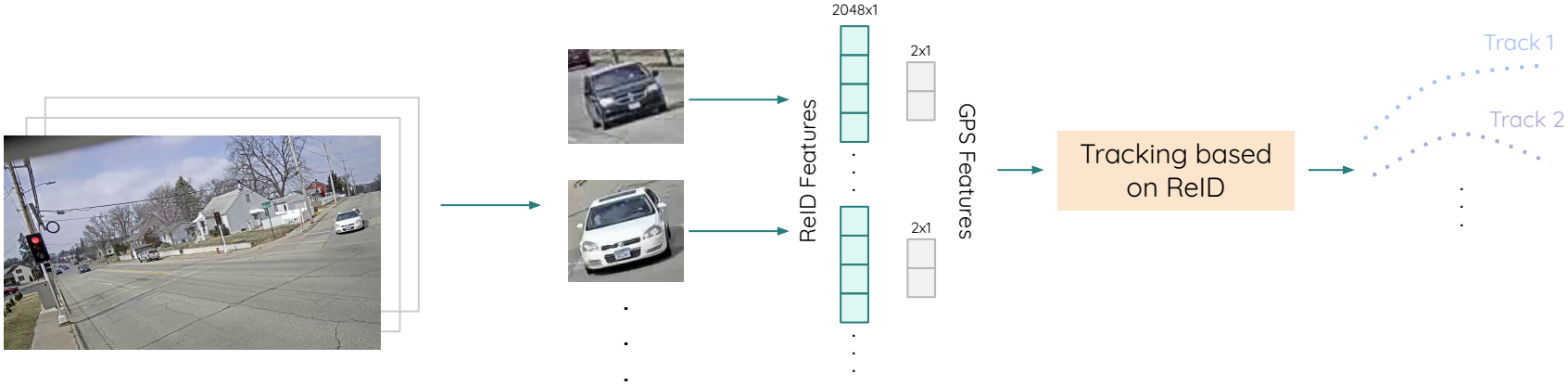
Repeated ids at the same camera and time instant

T2: Multi-Target Multi-Camera (MTMC) Tracking

Challenge Approach

In paper[2], it has been proposed a vehicle-reidentification with spatio-temporal consistency.

1. Use vehicle re-identification(ReID) model to:
   a. Detect vehicles and **compute GPS** information.
   b. **Extract ReID features** for each cropped image using a CNN.
   c. Obtain single tracks.



[1] https://github.com/didi/mtmc-vt
[2] Li, Peilun, et al. (2019, May). Spatio-temporal Consistency and Hierarchical Matching for Multi-Target Multi-Camera Vehicle Tracking. In CVPR Workshops (pp. 222-230).

T2: Multi-Target Multi-Camera (MTMC) Tracking

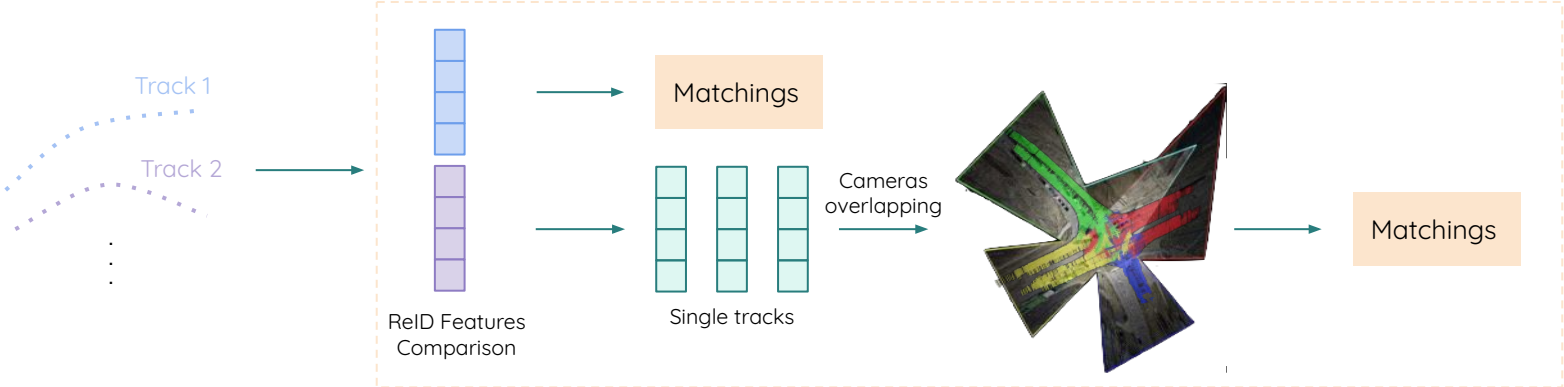
2. To do Multi-Target Multi-Camera Tracking:

   a. **Match tracks based on ReID features** (2 matching tracks are required from different cameras): merging tracks with high similarity of ReID features.

      i. Crossroad scenes: Car should be captured by different cameras at the same time + the direction should be consistent.

      ii. Arterial road scenes: A consistency only in the moving direction is required.

   b. **Post processing:** Those tracks that remain single are processed and the matching threshold is improved. This matching is done using overlapping information.

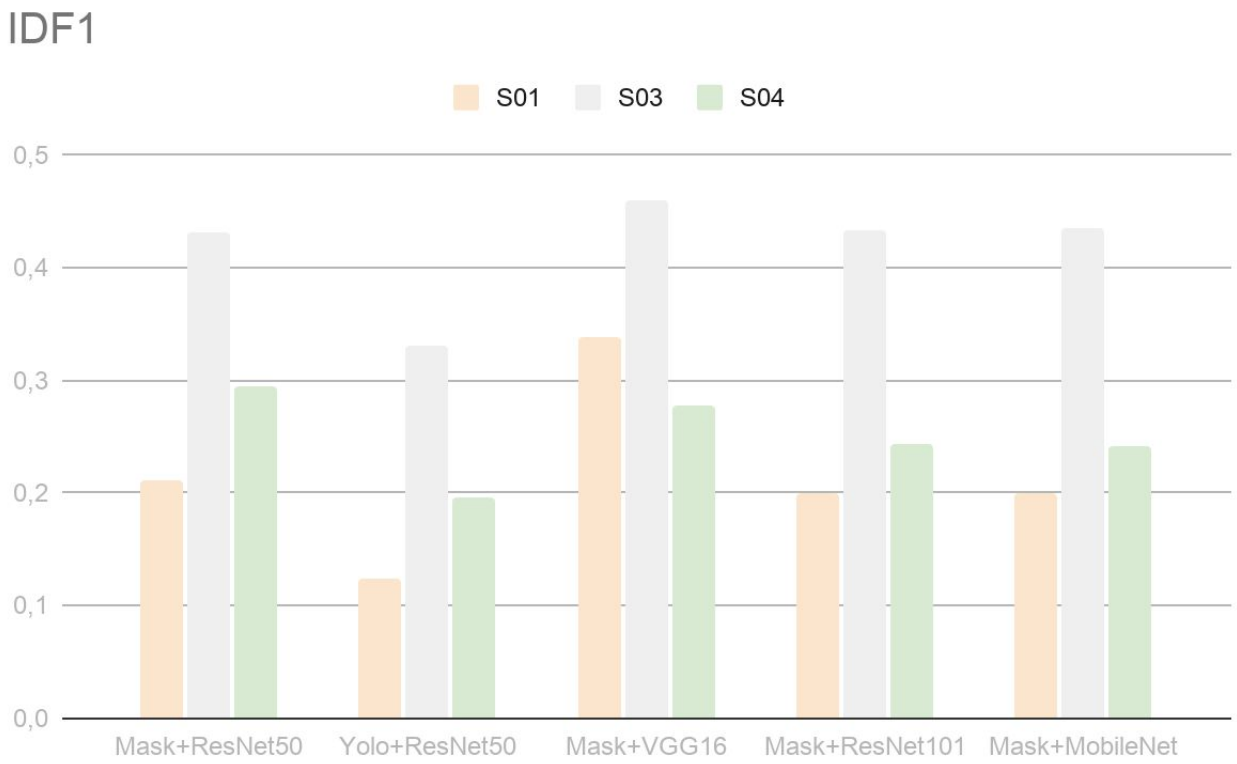


T2: Multi-Target Multi-Camera (MTMC) Tracking

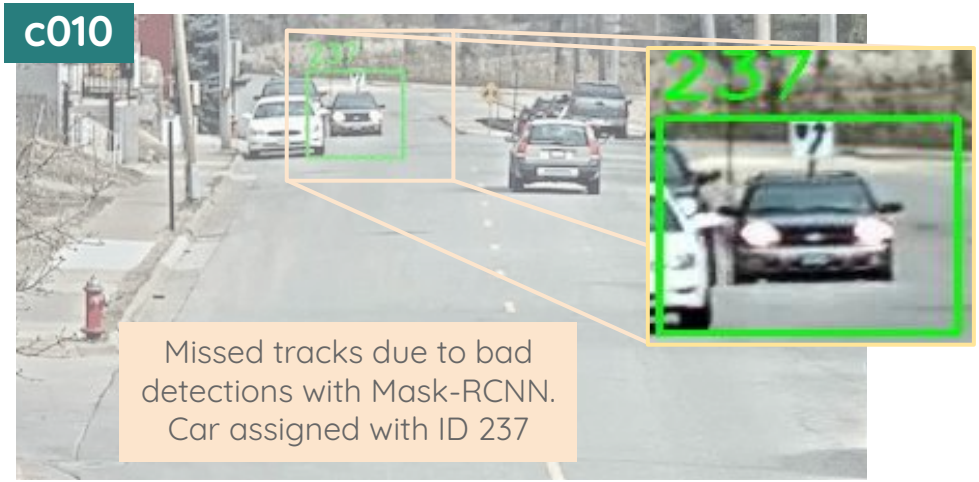
Configuration	Scene	IDF1	IDP	IDR	Precision	Recall
Baseline[1]: Detections with Mask-RCNN and car features with ResNet50 trained with Triplet Loss	**S01**	0.2112	0.236	0.203	0.4647	0.4358
	S03	0.4318	**0.33**	0.6417	0.387	0.7703
	S04	**0.2951**	**0.2246**	**0.487**	0.3478	0.7463
Detections with YoloV3 (trained by us) and features with ResNet50	**S01**	0.1248	**0.2828**	0.0876	**0.5647**	0.1979
	S03	0.3314	0.3073	0.3597	0.3653	0.4275
	S04	0.1968	0.1973	0.1962	**0.3954**	0.3932
Detections with Mask-RCNN and features with VGG16	**S01**	**0.3377**	0.2654	**0.4639**	0.4385	**0.7664**
	S03	**0.4599**	0.3157	**0.8467**	0.3446	**0.9243**
	S04	0.2778	0.1946	0.4851	0.326	**0.8126**
Detections with Mask-RCNN and features with ResNet101	**S01**	0.1992	0.2083	0.1908	0.4452	0.4079
	S03	0.4334	0.318	0.6799	0.3848	0.8226
	S04	0.243	0.1771	0.387	0.3198	0.6987
Detections with Mask-RCNN and features with MobileNet	**S01**	0.2002	0.2086	0.1924	0.4453	0.4107
	S03	0.4349	0.3235	0.6631	**0.3921**	0.8036
	S04	0.242	0.1782	0.3767	0.3245	0.6858

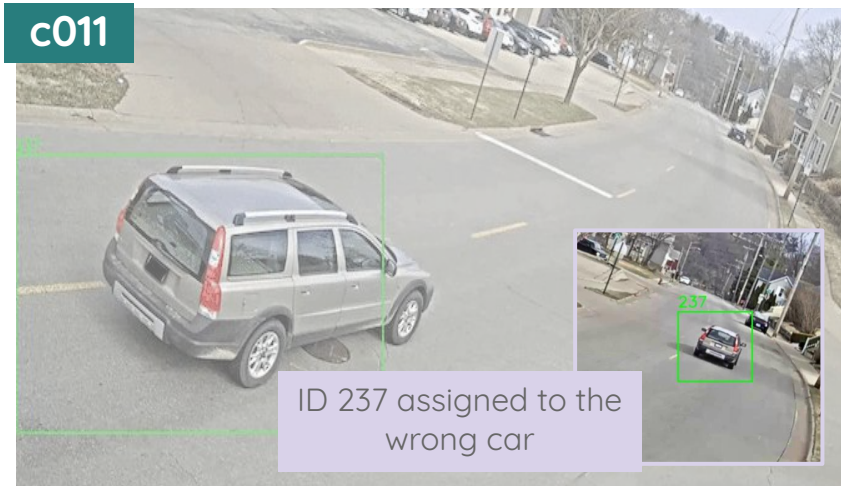
[1] https://github.com/didi/mtmc-vt

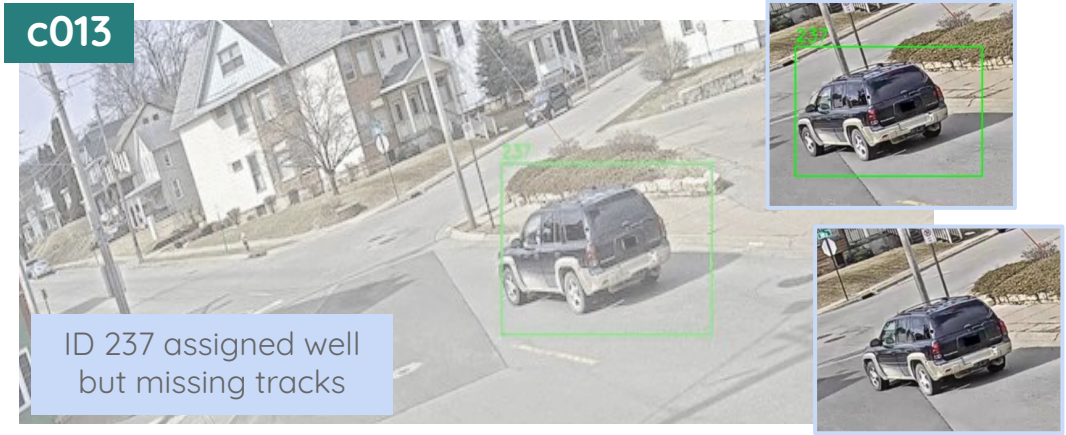
T2: Multi-Target Multi-Camera (MTMC) Tracking



T2: Multi-Target Multi-Camera (MTMC) Tracking







Conclusions

1. Depending on the Sequence processed, the methods that works better changes. However, looking on the computational cost, OF increases it a lot, not making it viable for use in Real time applications.
2. The limitations of image sensors are clear. Although there is still room for improvement, there are situations where would be necessary to include more information to improve the results. e.g. GPS, Radar, Lidar…
3. There is no one method that is the clear winner. Each situation requires a specific methodology that can maximize results. In doing so, one must also avoid overfitting.
4. MTMC is a challenging task:
    a. Tested hand-crafted techniques are way too simple.
    b. Spatio-temporal Consistency and Hierarchical Matching: Due to the variety of situations, different techniques must be applied depending on the scene, cameras, etc.

Q & A