Master in Computer Vision Barcelona

Project Module 6 Coordination

Video Surveillance for Road Traffic Monitoring

J. Ruiz-Hidalgo / X. Giró

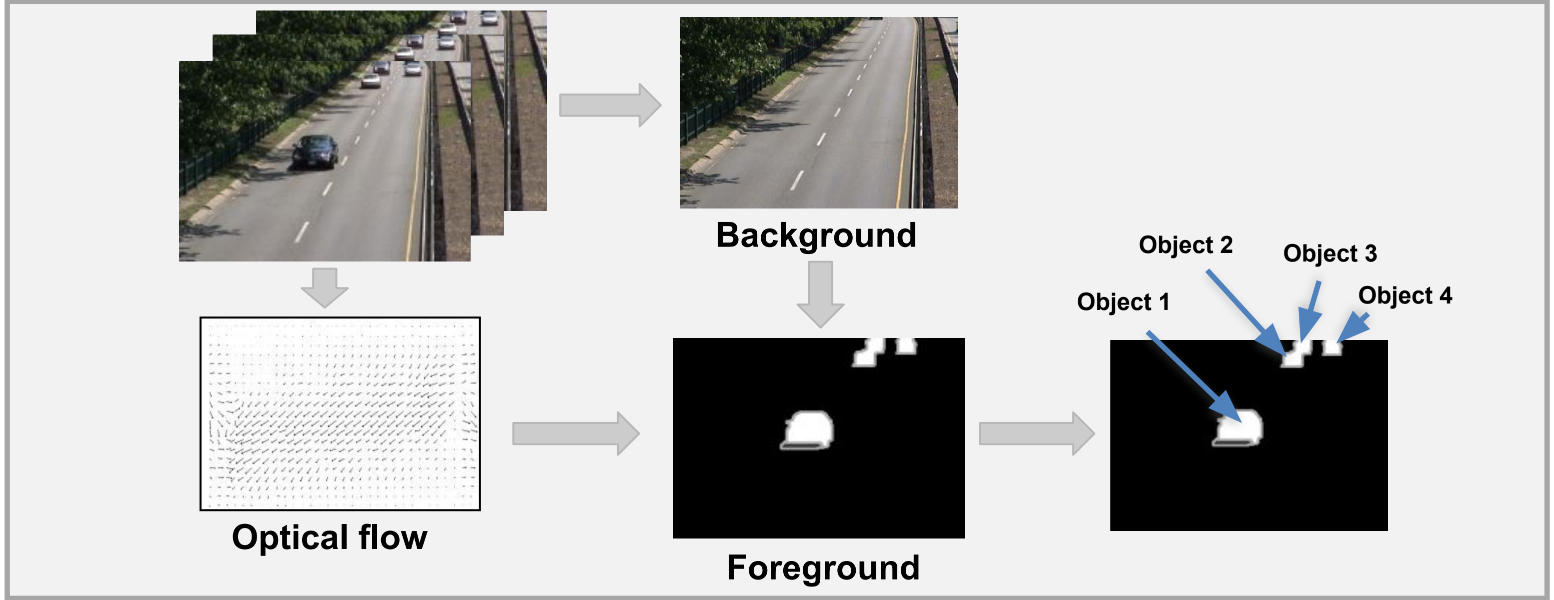
j.ruiz@upc.edu / xavier.giro@upc.edu

Week 3: Report

Teams & Repos (please state any change)

Repo	Students
Team 1	Rachid Boukir, Josep Bravo, Alex Martín, Guillem Martinez, Miquel Romero
Team 2	Álvaro Budria, Alex Carrillo, Sergi Masip, Adrià Molina
Team 3	Albert Barreiro, Manel Guzmán, Jiaqiang Ye Zhu and Advait Dixit
Team 4	Julia Ariadna Blanco Arnaus, Marcos Muñoz González, Abel García Romera and Hicham El Muhandiz Aarab
Team 5	Razvan-Florin Apatean, Michell Vargas, Kyryl Dubovetskyi, Ayan Banerjee and Iñigo Auzmendi
Team 6	Guillem Capellera, Ana Harris, Johnny Núñez, Anna Oliveras

Project Schedule



Week 1
- Introduction
- DB
- Evaluation metrics

Week 2
- Background estimation
- Stauffer & Grimson

Week 3
- Object Detection
- Tracking

Week 4
- Optical flow
- Tracking

Week 5
- Multiple cameras
- Speed

Week 6
- Presentation workshop

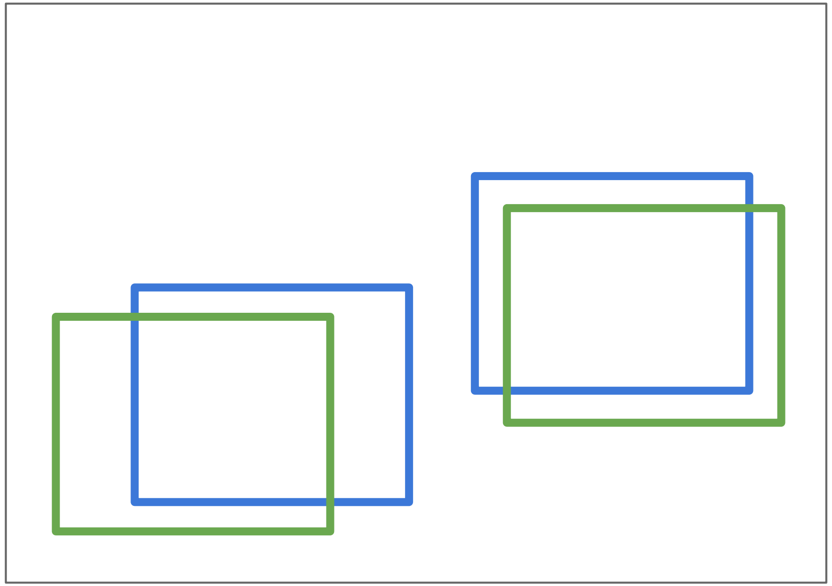
Tasks

- Task 1: Object detection
- Task 2: Object tracking
  - **Task 2.1: Tracking by Overlap**
  - Task 2.2: Tracking with a Kalman Filter
  - Task 2.3: IDF1,HOTA scores

Task 2.1: Tracking by Maximum Overlap

Basic algorithm (your task is to modify / improve it based on your experiments):

1. Assign a unique ID to each new detected object in frame N.
2. Assign the same ID to the detected object with the highest overlap (IoU) in frame N+1.
3. Return to 1.



Detected Objects @ Frame N

Detected Objects @ Frame N+1

Task 2.1: Tracking by Maximum Overlap (Team X)
- Copy & paste (max 2, pages)

Task 2.1: Tracking by Maximum Overlap (Team 1) [1/2]

The results reported in this section are based on the detections provided by **YOLOv8**, **fine-tuned** using the **first 25% of the dataset for training.**

Tracking by maximum overlap:

When comparing bounding boxes between frame N and frame N+1, two scenarios need to be considered. In the first frame, each object in the scene is assigned a different track. In the following frames, each bounding box in frame N+1 is compared with the bounding boxes in frame N.

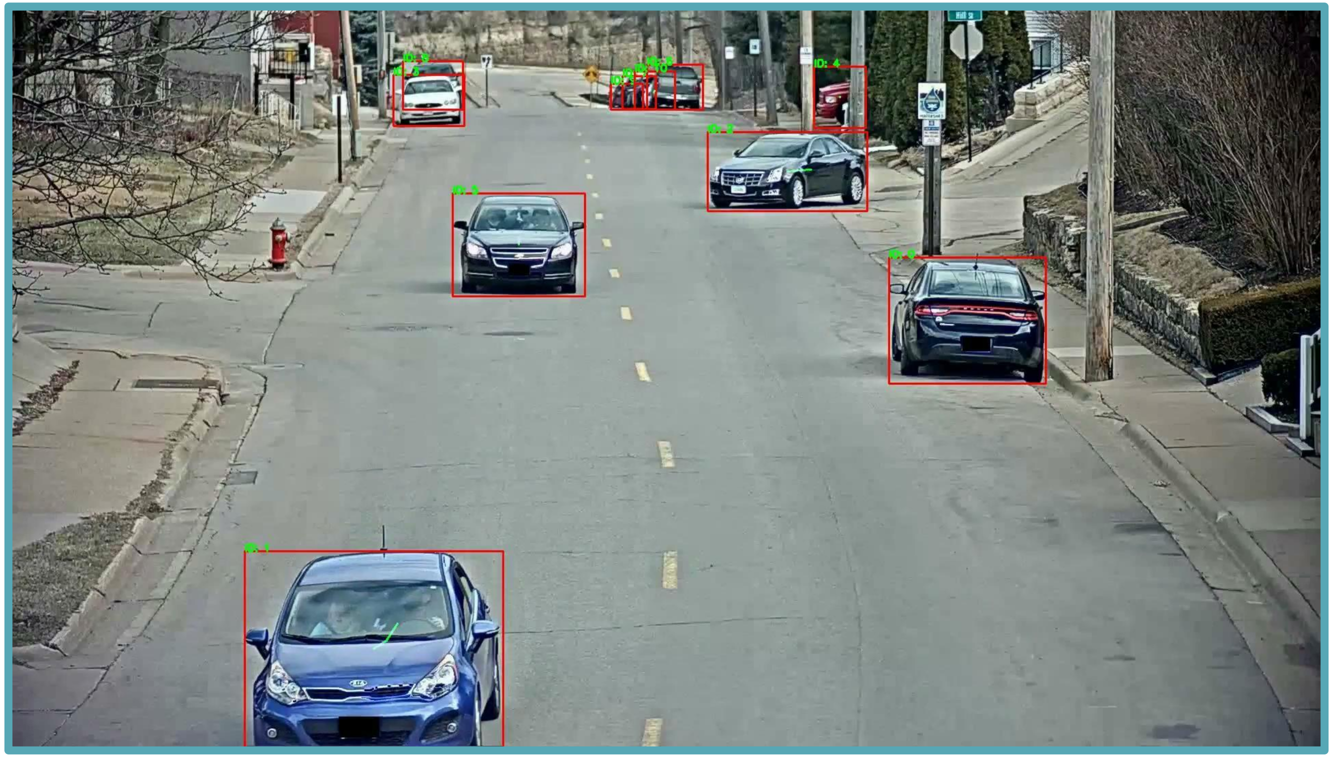
- If a bounding box is already labeled in frame N, the corresponding bounding box in frame N+1 is assigned the same track value. There is a **minimum overlap of 0.5 IoU** in order **to consider a candidate** track update. The **candidate** with **maximum overlap is assigned**.
- If a new bounding box in frame N+1 has no corresponding bounding box in frame N, it is assigned a new track label.

This is a **rather simple algorithm but useful**, and **enough** in most cases.

However, it has some **pre-conditions** (not very fast moment nor low frame rate) and **drawbacks** (overlaps between instances can cause confusion). Qualitative results in the next slide.

Detections	mAP (0.5) w/o confidence score	mIoU
Original Detections	0.89	0.84
IoU Overlap	0.89	0.83

Task 2.1: Tracking by Maximum Overlap (Team 1) [2/2]



- **Overlapping** might generate the assignation of **wrong ID**
- For instance, on the **0:12 of the video** we can see the moving car on the fron creating many new IDs.
- The **quality** of the **tracking directly depends on the** quality of the **bounding boxes**. We got very good boxes overall, that's why we do not see major problems besides last point.

Task 2.1: Tracking by Maximum Overlap (Team 2) [1/3]

At each frame of the video, the algorithm compares the current bounding box with the ones in the previous frame and assigns a track label based on the maximum overlap (IoU) between the bounding boxes.

Preprocessing:

- Filter detections by a confidence threshold. This makes the tracking more robust by using more confident detections.
- Apply non-maxima suppression to detections. This helps to improve the accuracy of the tracking by avoiding duplicate detections.

Algorithm steps:

1. In the first frame of the video, a different track label is assigned to each object in the scene.
2. **For each frame** after the first frame, the algorithm compares the bounding boxes of the current frame with the ones in the previous frame to assign a track label.
    - 2.1. **If** the bounding box in the current frame matches a bounding box in the previous frame (IoU ≥ Min IoU), the same track label is assigned to it.
    - 2.2. **Else**, if the bounding box in the current frame does not match any in the previous frame, a new track label is assigned to it.

We keep a history of the **live tracks** and the **terminated tracks**, and we add an additional hyperparameter to the algorithm:

- Max frame(s) skip: Maximum number of frames to skip before terminating a track. Thus,
    - **If** the *(current_frame_number - last_detected_frame_number) <= max_frame_skip*,
        - we add the track to the live tracks
    - **Else**,
        - we add the track to the terminated tracks

We test the Tracking by Maximum Overlap algorithm with the following set of **(hyper) params**:

- (**Pre-trained**) model	["yolo", "retina"]
- Min IoU	[0.5, 0.75]
- Max frame(s) skip	[5, 10]
- Confidence threshold	[0.0, 0.4]

with all the experiments filtering the detections by confidence threshold and applying NMS (iou_thr=0.7) always.

Finally, we test the algorithm with the detections inferred by our **finetuned Faster R-CNN** model and compare it with the pre-trained Faster R-CNN, as shown in table below.

The HOTA and IDF1 performance metrics are calculated using the TrackEval package.

Model	Confidence threshold	Min IoU	Max frame(s) skip	HOTA	IDF1
Faster R-CNN (finetuned, mAP .5600)	-	.50	10	**.693**	**.637**
Faster R-CNN (pre-trained, mAP .5316)	-	.50	10	**.608**	**.555**

Task 2.1: Tracking by Maximum Overlap (Team 2) [2/3]

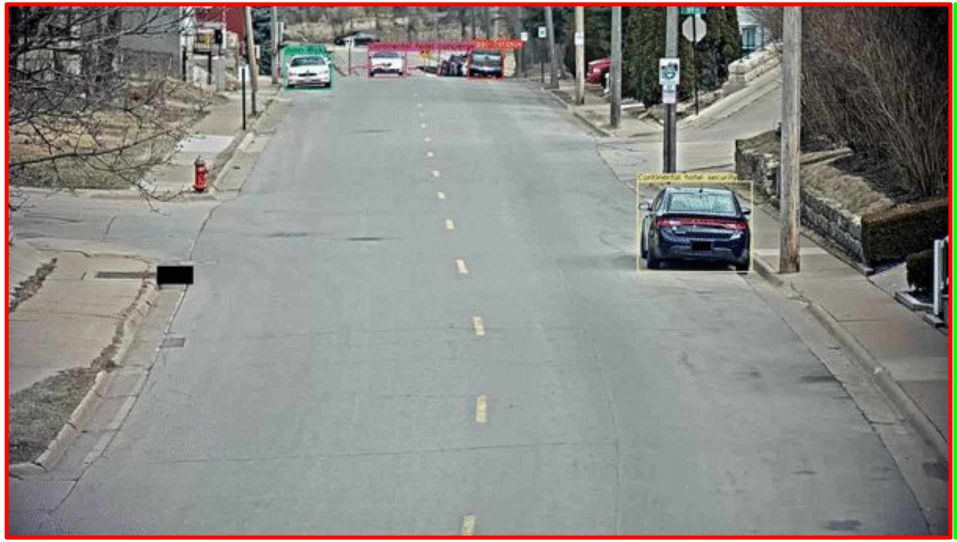
Model	Confidence threshold	Min IoU	Max frame(s) skip	HOTA ↓	IDF1
Retina (pre-trained, mAP .5312)	-	.50	10	**.597**	**.569**
	.40	.50	10	**.597**	**.569**
	.40	.50	5	**.595**	**.558**
	-	.50	5	**.595**	**.558**
	.40	.75	10	**.581**	**.536**
	-	.75	10	**.581**	**.536**
	-	.75	5	**.579**	**.531**
	.40	.75	5	**.579**	**.531**

Model	Confidence threshold	Min IoU	Max frame(s) skip	HOTA ↓	IDF1
YOLO (pre-trained, mAP .3636)	-	.50	10	**.576**	**.549**
	-	.50	5	**.576**	**.551**
	-	.75	10	**.532**	**.467**
	-	.75	5	**.531**	**.465**
	.40	.50	10	**.520**	**.523**
	.40	.50	5	**.507**	**.505**
	.40	.75	10	**.506**	**.501**
	.40	.75	5	**.497**	**.488**

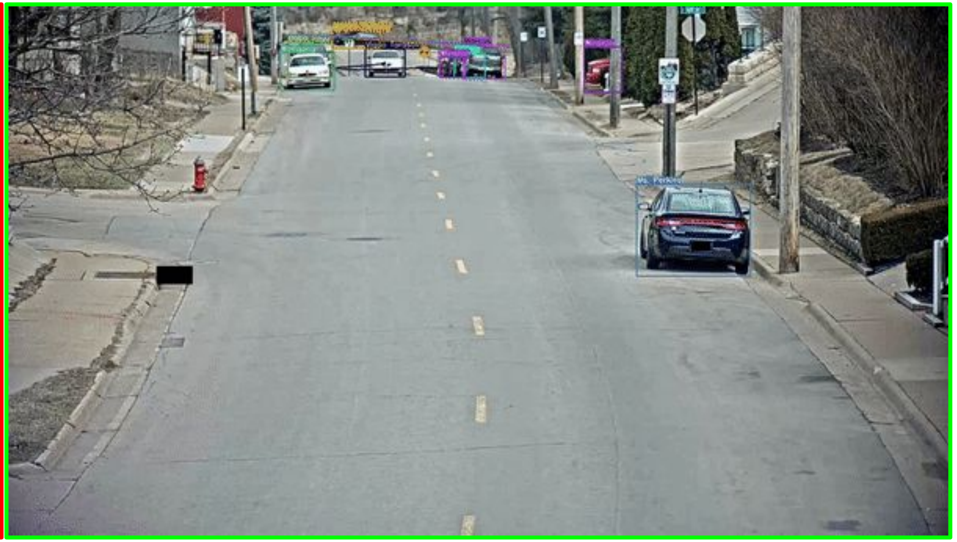
In the case of the pre-trained Retina model, the confidence threshold does not affect at all in the metrics (HOTA and IDF1), and a conservative IoU threshold of 0'50 performs better than increasing it to 0'75. Regarding the maximum frame skip, using a value of 10 slightly improves the performance, but it's not significant enough. In the case of the pre-trained YOLO model, there are significant improvements in terms of performance if no confidence threshold is applied. Again, an IoU threshold of 0'50 performs better, and now a maximum frame skip of 10 is always better than using 5 frames.

NOTE: The accuracy of the tracking is directly proportional to the accuracy of the bounding boxes detections.

Task 2.1: Tracking by Maximum Overlap (Team 2) [3/3]



YOLO (pre-trained)



Faster R-CNN (finetuned)

Model	Confidence threshold	Min IoU	Max frame(s) skip	HOTA	IDF1
Faster R-CNN (finetuned, mAP .5600)	-	.50	10	**.693**	**.637**
YOLO (pre-trained, mAP .3636)	.40	.75	5	**.497**	**.488**

Here we show our best results with the worst ones, to clearly see a qualitative comparison.
In **YOLO**, new object instances that enter the scene suffer from a flickering effect, causing the tracking to continuously assign a new identifier until the object's bounding boxes stabilizes. This is also the case when the object exit the scene.
However, in our **finetuned R-CNN** this problem is not seen anymore; nor at the entrance nor exit of objects into the scene.
By plotting the trace of each tracking, we note that **Faster R-CNN provides more continuous and straight tracking lines**, while in YOLO the lines stumble a bit.

Task 2.1: Tracking by Maximum Overlap (Team 3) 1/2

Algorithm:

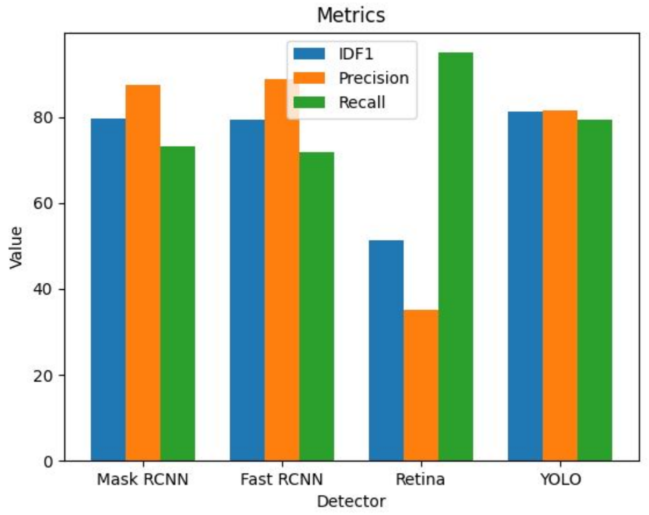
- Collect all the bounding boxes for each frame obtained using the trained model.
- Loop over all the frames, excluding the last one. Assign a new ID to each unassigned block.
- For each track, compute IOU for each detection for next frame.
- Set threshold of IoU ≥ 0.5 to determine the correspondence of consecutive bounding boxes
- Assign the bounding box to the closest track which has the maximum overlap or the maximum iou.

Cars that are far from the camera or have overlap(cross paths with others) or occluded are difficult to track and confuse the model

Cars that are close to the camera and well separated are easy to detect and the model can track them properly

Task 2.1: Tracking by Maximum Overlap (Team 3) 2/2

Detector	IDF1	IDP	IDR
Mask RCNN	79.8	87.55	73.17
Faster RCNN	79.45	88.8	71.89
Retina net	51.28	35.1	95
YOLO(Fine tuned)	**80.4**	**81.5**	**79.47**



- The accuracy of the model makes a huge difference to the accuracy of tracking. If there are false detections or missed detections, the model is not able to track the cars properly and the accuracy reduces.
- The Iou reduces and fluctuates for the cars which are far away or are occluded and the tracking is lost as the model confuses it with another car. The cars near to camera are well detected and tracked properly.
- Overlap Tracking is a simple and fast model but not very efficient and accurate, to overcome these difficulties we move on to Kalman Filters.

Task 2.1: Tracking by Maximum Overlap (Team 4) [1/2]

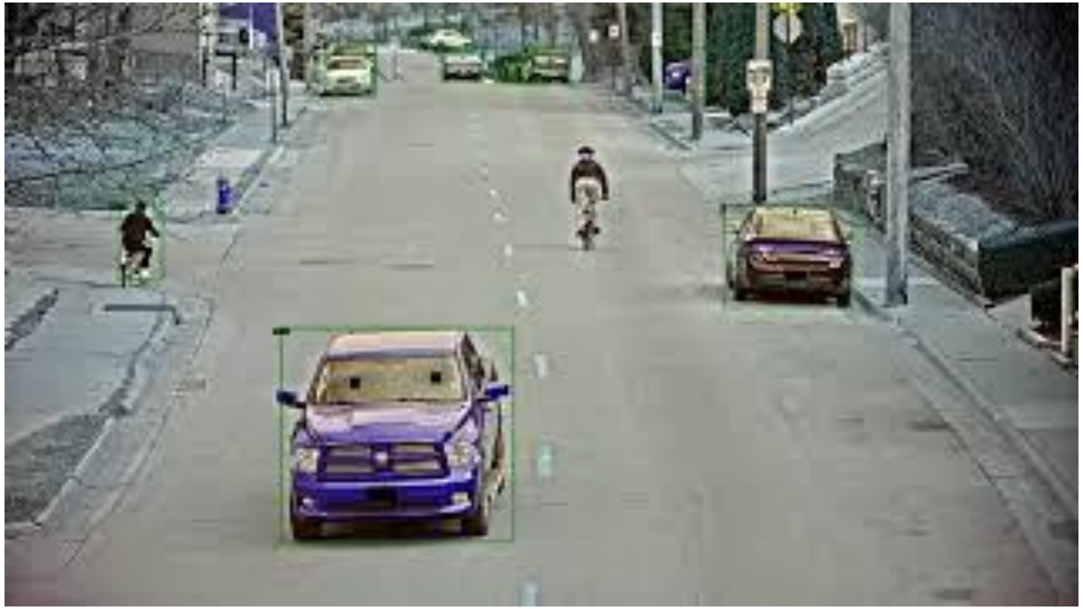
Algorithm used

1. Detect objects
2. Discard objects with confidence score lower than 0.5
3. Assign the object to an existent or a new track depending on the scenario:
    a. If we are in the first frame, assign each object to a different track
    b. In the following frames, compare the bounding boxes of the objects detected in the frame N+1 with the bounding boxes in the frame N.
        i. Pick the candidate with the maximum overlap, and if the IoU is greater than 0.4, assign the detected bounding box to the corresponding track ID of frame N
        ii. If the candidate with the maximum overlap doesn't have a IoU greater than 0.4 or no candidate has been found, assign the object detected to a new track

Model	IDF1	HOTA
Mask R-CNN X101-FPN off-the-shelf (Detectron2)	52.695	58.224
RetinaNet R101 off-the-shelf (Detectron2)	48.736	51.557
Faster R-CNN X101-FPN off-the-shelf (Detectron2)	44.382	55.636
Faster R-CNN fine-tuned	**76.938**	**80.878**

The accuracy of the model is crucial to improve the performance on tracking

Task 2.1: Tracking by Maximum Overlap (Team 4) [2/2]



Overlapping bounding boxes cause the algorithm to assign a new track ID to the same object. This can be observed in the cars parked farthest to the camera.

Task 2.1: Tracking by Maximum Overlap (Team 5)[1/2]

The main idea of overlap maximization algorithm implies comparing bounding boxes between frame N and N+1.
Initially, each bounding box in the first frame is given an ID. Then, based on IoU, we assign an existing ID or create a new one for the bounding boxes in the following frames.

Algorithm used is straight forward:

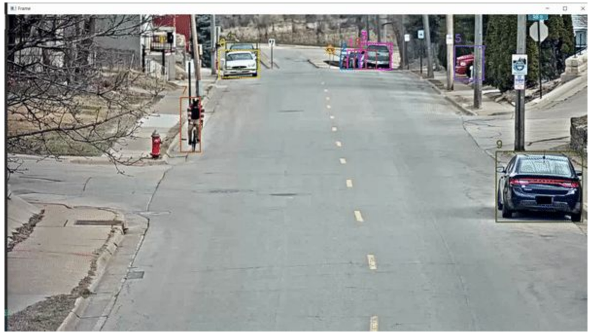
- Get the bounding boxes from the current frame
- Compare with the detected bounding boxes from the last 5 frames. By doing this, we can avoid situations where an object is identified differently due to sudden non-detections of objects.
- For each bounding box in current frame we get the bounding box from past frames that have the highest IoU that is bigger than 0.8.
- If there are no bounding boxes in past frames that match these conditions, a new ID is given to the object in current frame.

Observations:

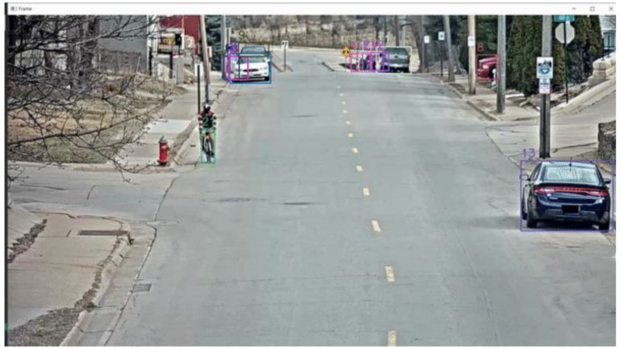
- Overlapping objects and fast moving objects (close to the camera moving cars) are continuously assigned a new identifier
- For an IoU threshold that is smaller (ex: 0.3-0.5) there are lesser identifications, which translates in better following of objects. However, this could make to assign the same id to a different object when a small overlap of the objects occur. This could be fixed using Kalman filters.

Task 2.1: Tracking by Maximum Overlap (Team 5)[2/2]

Here is a video with the bounding boxes when the IoU threshold is 0.8 and we are comparing with the last 5 frames.



Using Fine-tuned DETR



Using pretrained DETR

As can be seen, a robust object detector can significantly improve the performance of an object detection method, since it provides more precise IoU values for the detected boxes and ensures consistent object detection.

Maximum Overlap	IDF1	HOTA
Fine tuned DETR	**85.7**	**83.07**
Pretrained DETR	33.95	43.43

Task 2.1: Tracking by Maximum Overlap (Team 6) [1/3]

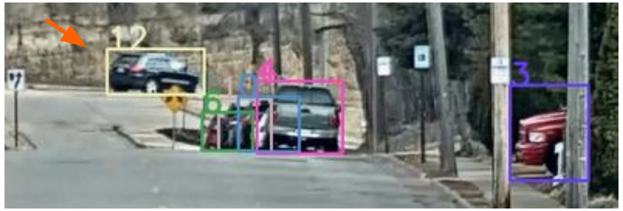
Before tracking, boxes with an overlap greater than 0.9 were excluded. In addition, we implemented a filter to remove the bounding boxes that detected bikes, since our model was trained only for cars, being this one of the limitations during tracking. This filter discarded those bounding boxes with a value greater than 230 on the y-axis and with a ratio (h/w) greater than 1.2

Maximum Overlap algorithm:
We compare the maximum overlap between bounding boxes of frame N to frame N_{n+1}.
We took into account the following conditions:

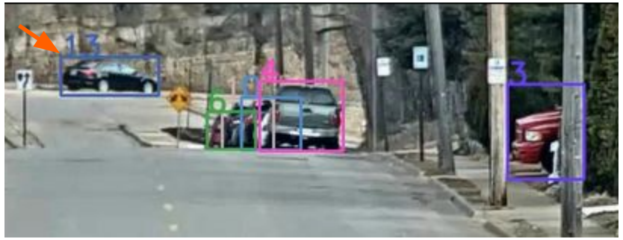
- In frame N_0 we initialize all the IDs of the objects present at N_0.
- When comparing objects present in N_{n+1} with those present in N, we set a threshold of iou>0.5.
- If there is no maximum overlap above 0.5, a new ID is added to that object. The same for those boxes with maximum overlap with the same object in the previous frame.
- We added a memory function that considered static and moving elements separately. So, if they appeared consecutively, their IDs were kept, if not, we dropped them. This was done with a frequency of 5 frames.

One of the main shortcomings of this method is the interference generated by the cars behind, in addition to the memory effect, this will be corrected with Kalmann later. In this example we see how a car labeled as 12 passes, and a few frames later another one with ID 13 passes, but when it approaches the point where there are more bounding boxes and the detection becomes smaller due to the occlusions of the image itself, this object is again labeled as 12, despite being a different car.

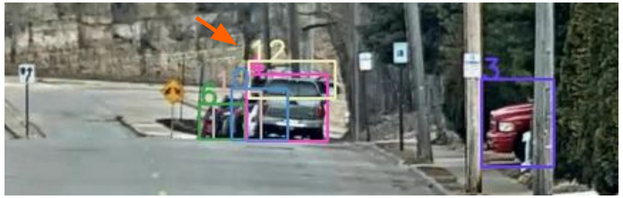
Frame N: FasterCNN



Frame N_{+1}



Frame N_{+2}



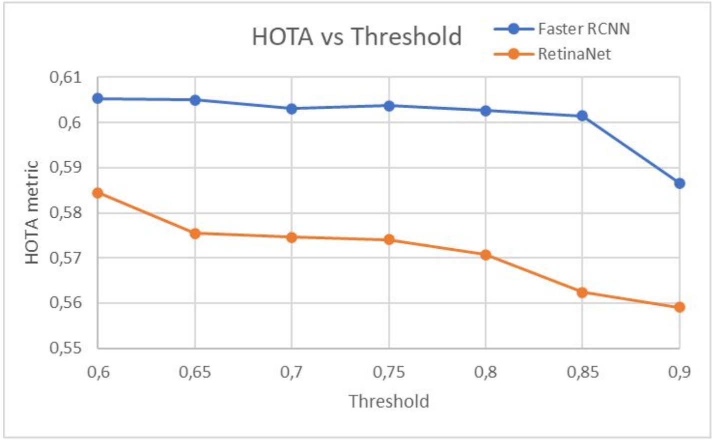
Task 2.1: Tracking by Maximum Overlap (Team 6) [2/3]

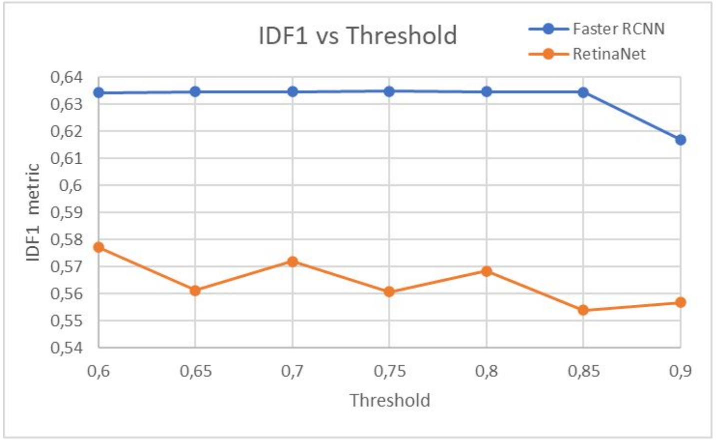
Multi-object tracking (MOT) metrics from TrackEval repository [1] were used to evaluate our analysis. IDF1 considers the association between the set of GT tracks and the predicted tracks to determine the trajectories, and is used in addition to another more restrictive metric, with IDF1 being a secondary metric. We therefore use HOTA, which, in addition to accounting for detection and association errors, takes into account the localization accuracy.
We performed a grid search with different confidence values, so that the boxes with a confidence lower than the threshold were removed. This was done for both FasterRCNN and RetinaNet.

Best results with Faster RCNN

confidence: 0.6

***Results with parked cars included**



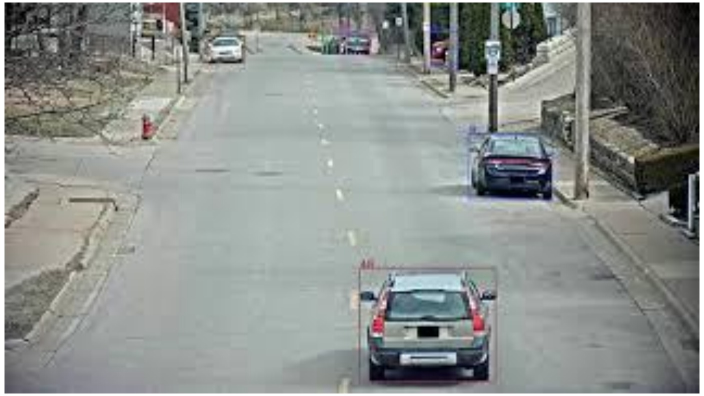


- The results obtained show greater stability in the case of faster than retina.
- In general, the metrics decrease when we increase the confidence, since we are discarding many more bounding boxes.
- We can see reflected in RetinaNet (IDF1) the instability mentioned in the previous slide, due to the low memory of the algorithm when associating objects throughout the frames.

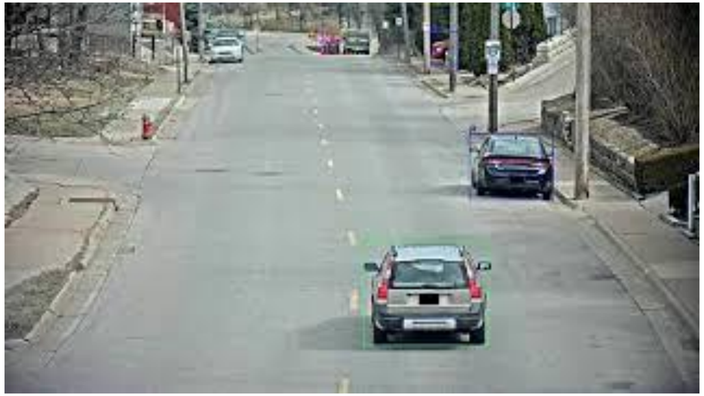
Task 2.1: Tracking by Maximum Overlap (Team 6) [3/3]

In this slide we compare the performance of both networks with a **confidence** value of **0.8**. Although the highest scores were obtained from a confidence value of 0.6, if we increase the confidence we are restricting the presence of incorrect detections.

Faster RCNN



RetinaNet



- Although we use a bounding box filter for bikes, some wrong detections are still present in both networks as expected.
- The background problem occurs equally for both networks. However, Faster RCNN seems to have a more stable behavior in general, since small bounding boxes do not appear randomly on others. As much as it happens in the case of the retina.
- In conclusion, this method is effective for tracking objects that are not occluded by others, since they can adopt the ID of those that overlap them by mistake, or that do not move at high speed over short distances, considering that the algorithm is not able of updating the memory correctly in these cases.

Task 2.1: Feedback

	feedback Implementation?
Team 1	Qualitative and quantitative results provided. why do you use mAP and not IDF1 or HOTA?? Highest overlap on IoU (threshold of 0.5) Good discussions.
Team 2	Qualitative and quantitative results provided (IDF1 0.637 with FRCNN) Highest overlap on IoU (threshold of 0.5) Good discussions Results provided for other models as well. Good.
Team 3	Qualitative and quantitative results provided (IDF1 80.4 with yolo) Good discussions Highest overlap on IoU (threshold of 0.5)
Team 4	Several models analyzed. Best FRCNN (IDF1 76,9) Quantitative results provided. No discussion of the results
Team 5	Qualitative and quantitative results provided IDF1 and HOTA (IDF1 85.7 with DETR) Good discussions Highest overlap on IoU (threshold of 0.8)
Team 6	Very good explanation of the process followed. Excellent analysis of the results. Plots of the metrics/vs threshold.

Tasks

- Task 1: Object detection
- Task 2: Object tracking
  - Task 2.1: Tracking by overlap
  - **Task 2.2: Tracking with a Kalman Filter**
  - Task 2.3: IDF1 score

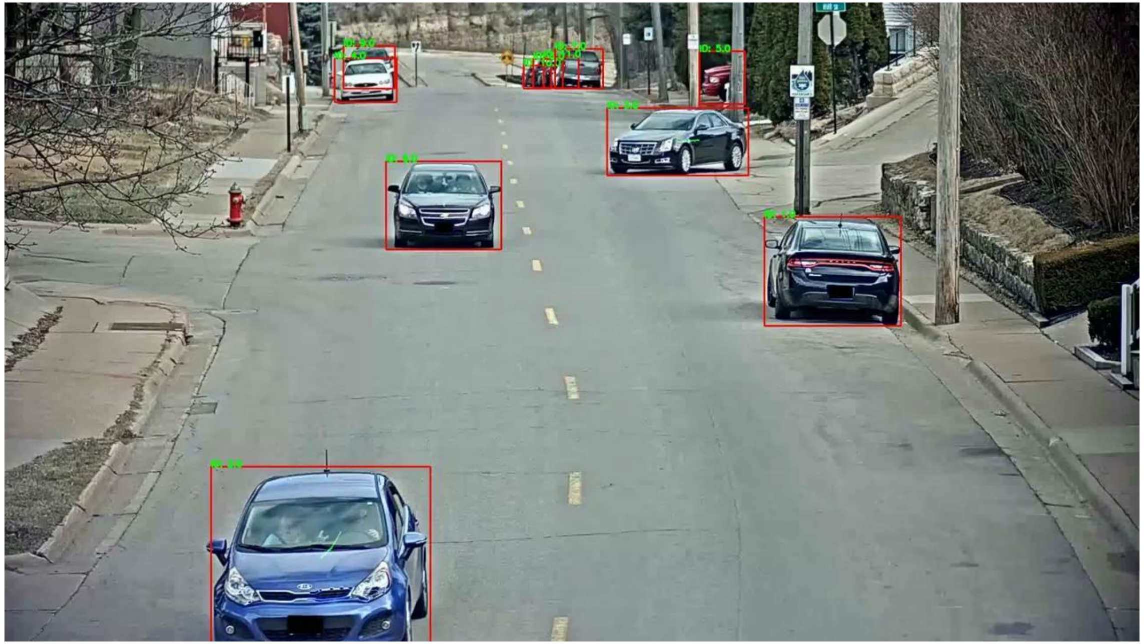
Task 2.2: Tracking with a Kalman Filter (Team X)
- Copy & paste (max 3 pages)

Task 2.2: Tracking with a Kalman Filter (Team 1) [1/3]

- The Kalman filter was implemented using the **S**imple **O**nline and **R**ealtime **T**racker presented by Alex Bewley in this paper. The github repository of the code can be found in the following link, where we used sort.py script in our project:
    - https://github.com/abewley/sort
- The sort tracker receives precomputed bounding boxes of each frame with or without confidence scores as input, and it returns them without the confidence score, with their corresponding ids and with slightly refined bounding boxes.
- SORT algorithm predicts the position of each bounding box from one frame to the next one. Then, the bounding boxes are associated computing the IoU between the predicted bounding box after displacement, and the detected bounding boxes in the next frame. One of the parameters that can be defined by the user is the minimum IoU threshold that is imposed to reject assignments. In this case has been used IoU threshold of 0.3. It also has 2 more parameters that can be tuned:
    - Max_age: refers to the maximum number of frames that an object can be still tracked without finding any assignment above the min IoU threshold. Useful for occluded objects.
    - Min_hits: refers to the minimum number of necessary detections to confirm an object and start tracking it.
    - In our specific scene we did not see appreciable improvement by tuning that parameters, so we put the default values on them (max_age=1 and min_hits=3).
- The results obtained using our best model, YOLO v8 model fine-tuned in task 1.3 and trained using 4-fold cross-validation strategy B, are the following:

Detections	mAP (0.5) w/o confidence score	mIoU
Original Detections	0.89	0.84
IoU Overlap	0.89	0.83
Kalman Filter using SORT	0.88	0.82

Task 2.2: Tracking with a Kalman Filter (Team 1) [2/3]



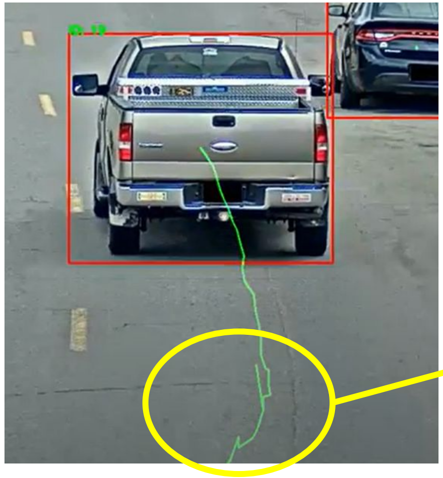
- The tracking using Kalman Filter is smooth but we still have the overlapping problems where new IDs are created for the same car, for example the red car in the second 14 of the video. This could be solved by increasing the min_hits value, as the second bounding box appear only in a few frames, but by increasing the min_hits we would also lose track of cars passing by the street in the background that appear only in a few frames.
- On the other hand, in the second 14 we lose the tracking IDs of the occluded cars that are at the background parked on the right of the street. In this case this problems can be solved by increasing the max_age value, but we would have the consequences that the cars would be still tracked even when they are not in the scene anymore.

Task 2.2: Tracking with a Kalman Filter (Team 1) [3/3]

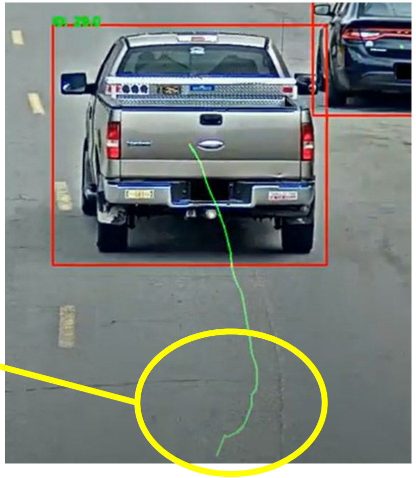
Discussion Maximum Overlap vs. Kalman Filter

- Observing the quantitative results we can see that:
  - The mAP(0.5) and mIoU (the tracking metrics will be evaluated in the next task), are nearly the same obtaining slightly better results using the Maximum Overlap tracking. This could be due to the fact that SORT algorithm approximates the inter-frame displacement of the cars with a linear constant velocity model.
- Observing the qualitative results we can observe that:
  - The behaviour of both tracking algorithms is very similar.
  - SORT algorithm is more smooth tracking the vehicles.
  - When a new vehicle appears in the scene, the Maximum Overlap tracker starts to track the vehicles some frames before the SORT algorithm. This is due to the min_hits parameters that starts to track an object after detecting it in a minimum of 3 frames.
  - The problem of overlapping IDs appears in both algorithms, but in SORT algorithm can be mitigated this behaviour.

Maximum Overlap Tracking



In the trajectory line can be observed the differences in smoothness between both algorithms



Kalman Filter using SORT algorithm

Task 2.2: Tracking with a Kalman Filter (Team 2) [1/3]

In this task, we perform tracking using Kalman Filters. We use the implementation from *SORT: A Simple, Online and Realtime Tracker*, which defines a **linear constant velocity model** with a **state variable x=[x, y, s, r, dx, dy, ds]** where (x, y) is the center of each bounding box, s is the scale and r is the aspect ratio, which is assumed to be constant throughout time. Therefore, only the velocities for the position and the scale are computed.

SORT method steps:

1. Optimally predict next bounding box (bb) location for each track object using Kalman.
2. Assign each observed bb to a track bb using the Hungarian algorithm (*min IOU* required).
3. Perform the Kalman correction with the observed bbs.
   a. If the tracker is unmatched, do not correct.
4. Create new trackers for unmatched observed detections.
5. Remove trackers that were not updated for *max_age* time steps.
6. Return trackers with more than *min_hits* and repeat.

We test the SORT algorithm with the following set of (**hyper) params**:

- Confidence threshold	[0.5]
- min_hits	[3]

with all the experiments filtering the detections by confidence threshold and applying NMS (iou_thr=0.7) always.

In the following experiments, we evaluate the pre-trained and fine tuned versions of Faster RCNN on HOTA and IDF1 again. We experiment with several values for the *min_iou* and *max_age* parameters.

The ***min_iou*** parameter is used to decide whether to reject assignments where the observed bb overlaps less than *min_iou*.
The ***min_hits*** parameter sets a tracker to *latent[check speaker]* when it has been updated for at least *min_hits* in a row.
The ***max_age*** parameter tells the algorithm to remove a tracker that has not been updated for *max_age* steps.

Task 2.2: Tracking with a Kalman Filter (Team 2) [2/3]

To study the effect of the Min IOU and the Max age, we set a baseline represented in the first green row.

Model	Min IOU	Max age	HOTA	IDF1
Faster R-CNN (pre-trained, .5316 mAP)	0.3	10	**0.626**	**.591**
	0.1	10	0.62	.587
	0.5	10	0.621	0.583
	0.3	**1**	0.603	0.558
	0.3	**50**	0.626	0.590

Model	Min IOU	Max age	HOTA	IDF1
Faster R-CNN (fine tuned, 0.560 mAP)	0.3	10	0.723	.714
	0.1	10	0.717	.704
	0.5	10	0.691	0.668
	0.3	**1**	0.678	0.629
	0.3	**50**	**0.7428**	**0.784**

Although the authors of SORT state that the **Min IOU** handles short-term occlusion caused by passing targets, we do not observe a significant difference when tweaking it. It seems like the default value of 0.3 works the best.

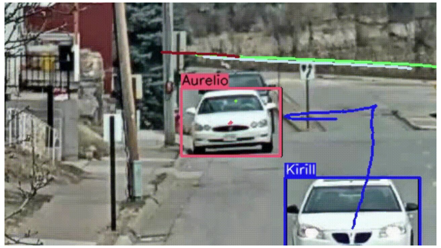
Regarding **Max age**, we can see that **lowering it punishes the performance in both cases**, **as it allows for a higher tolerance to occlusions or false negatives** generated by the detectors. However, setting it higher in the case of the pre-trained network doesn't affect the performance at all, while the fine tuned network yields a decent improvement. This may be a sign that **the performance of the pre-trained model is limited by its detection capabilities**. Therefore, we suspect that **improving the detection model would yield even better results**.

Finally, we see an increase of **5 points in the HOTA and 15 in the IDF1 metrics w.r.t. the best Maximum Overlap solution**. Since IDF1 focus more on the associativity rather than the detections accuracy, this large difference in the increases may be explained by the fact that **this Kalman filter technique allows for better handling of occlusions and consistency** between frames.

Task 2.2: Tracking with a Kalman Filter (Team 2) [3/3]

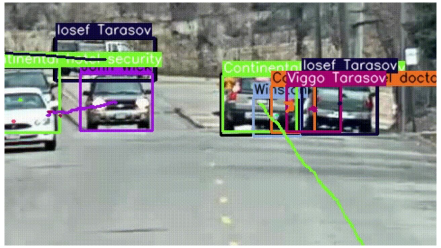
Maximum Overlap (best)
Faster R-CNN (finetuned)

Kalman Filter (best)
Faster R-CNN (finetuned)



The detector fails, but Kalman preserves the IDs of the trackers.

We compare the SORT method using Kalman Filter with our previous algorithm based on Maximum Overlapping. The main difference we perceive is that **Kalman Filter is able to maintain an ID consistency over time more effectively**, even when there are **occlusions or the detector fails to detect the object** (this is tuned by the *max_age* parameter). However, Kalman fails when there are **hard occlusions** and the car is moving (e.g. two cars intersect in opposite directions). In these visualizations, we are also displaying the raw detection (in color) and the updated tracking bounding box (in black). Notice how when a car enters the scene, the updated box noticeably differs from the raw detection, as **we are using a Linear Constant Velocity model**. Although it is harder to see, when the cars turn, the model also takes a bit longer to compensate.



Example of the robustness of Kalman against occlusions (please, ignore the ghost detection)

Task 2.2: Tracking with a Kalman Filter (Team 3)[1/3]

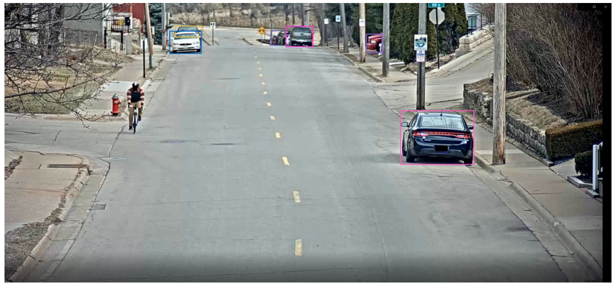
To perform tracking with Kalman filter we have used the SORT (Simple Online and Realtime Tracking) By Alex Bewley. We have used the suggested implementation.

In SORT the tracker receives precomputed bounding boxes of each frame and predicts their position in the next frame using a motion model. The IoU between predicted and detected bounding boxes is computed to associate them. SORT has tunable parameters, including minimum IoU threshold, max_age, and min_hits.
Where:

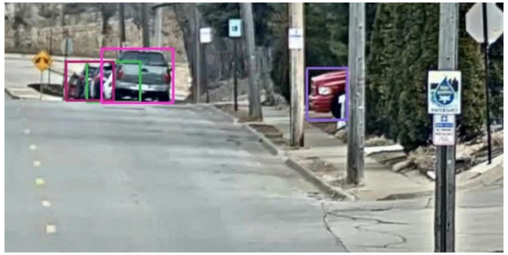
- IoU threshold: It is the minimum Intersection over Union (IoU) value between the predicted bounding box and the detected bounding box in the next frame for an assignment to be accepted.
- Max_age: It is the maximum number of frames that an object can be still tracked without finding any assignment above the min IoU threshold. This parameter is useful for occluded objects that are temporarily not visible. The default value for this parameter is 1.
- Min_hits: It is the minimum number of necessary detections to confirm an object and start tracking it. This parameter is used to ensure that only objects with a minimum number of detections are tracked. The default value for this parameter is 3.

We have check several combinations, however no one have give us more than 1% performance increase, so we used the default values for our scene.

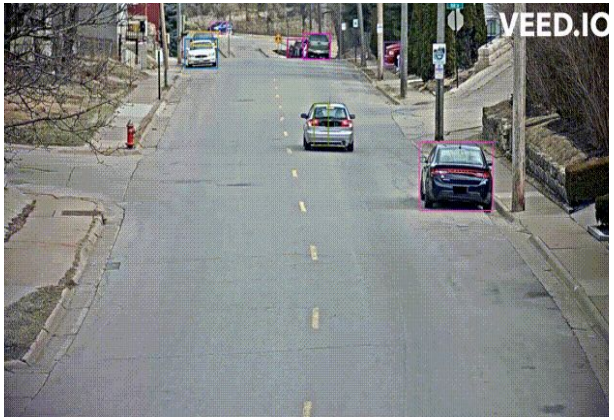
As for the maximum overlap case the evaluation of the tracker is heavily reliant on the object detector accuracy and precision. We have also used fine tuned YOLO bounding box predictions.



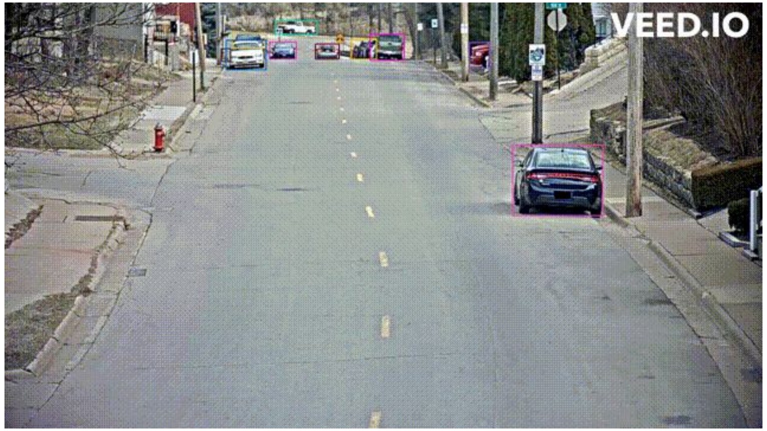
Task 2.2: Tracking with a Kalman Filter (Team 3)[2/3]



Performs well on occluded STATIC objects. Even when losing the track for seconds it is able to recover.



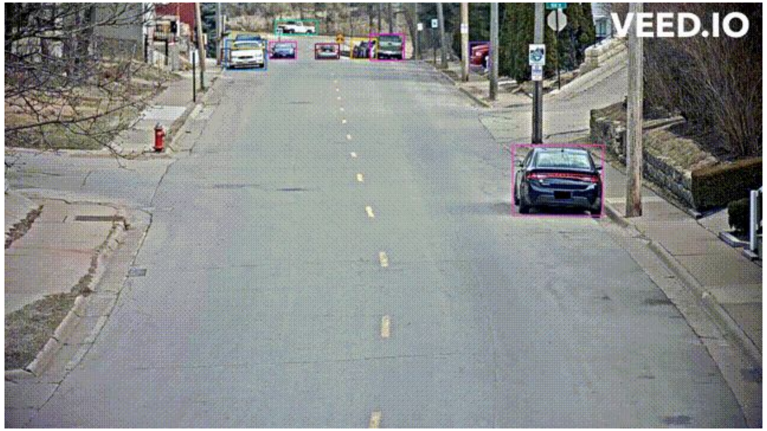
As we can see at the very background of the image the tracker struggles to follow the car that goes along the road and mix the boxes with the one parked



The overall performance is quite good, however if there are occlusion while the car is moving the track can be lost. The bounding box suffer from flicking until losing the id and create a new one.

Task 2.2: Tracking with a Kalman Filter (Team 3)[3/3]

Kalman Filter vs Maximum Overlap

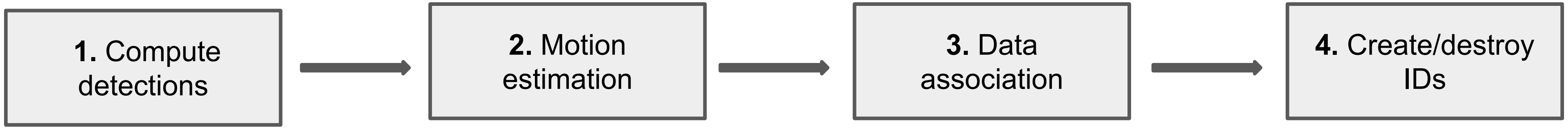


The Kalman filter get an overall best result. The boxes obtained are more consistent and resistant to change to a new id. Especially on occlusion where both models suffer big problems trying to keep the track.

Task 2.2: Tracking with a Kalman Filter (Team 4) (1/3)

Tracking using SORT

To improve our tracking results, we used Alex Bewley's implementation of Simple Online and Realtime Tracking (SORT) [1]. SORT is an algorithm that uses the Kalman filter for motion estimation, its pipeline is explained below:



1. The first step of the algorithm is to **obtain a set of non-tracked detections as a starting point** for the tracking stage. We have used FasterRCNN, RetinaNet and MaskRCNN only considering the car classes and using a threshold of 40% on the confidence.

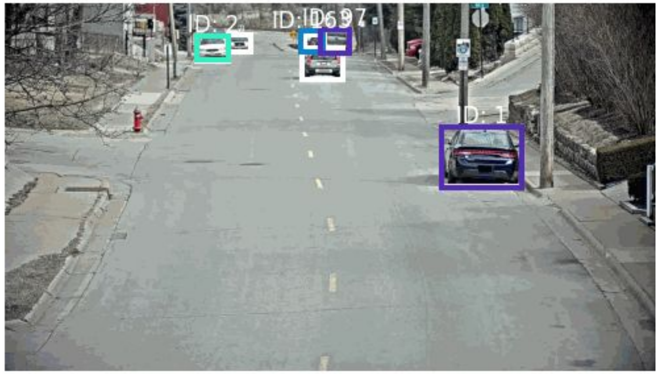
2. Detections between frames are propagated **using a linear constant velocity model with the Kalman filter**. Additionally, gaps in the object detection stage will also be predicted with the linear velocity model.

3. SORT uses a data association algorithm to associate the objects between frames. The cost matrix for **the assignment is generated by computing the intersection-over-union (IOU)** distance between each detection and all of the predicted bounding boxes for the current targets. In particular, the Hungarian algorithm is used to solve it. Besides, if the IoU is below a minimum, the box will get discarded.
4. New **trackers are created when a bounding box has a IoU under a IoU_{min} threshold** (e.g. car entering the scene). On the other hand, trackers will be destroyed if they do not appear in the image for at least T_{lost} frames. This means that the algorithm will fail to assign the same id to a reappearing object, as it will assume a new tracker the second time it enters the scene.

[1] Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2016). Simple online and realtime tracking. In 2016 IEEE International Conference on Image Processing (ICIP) (pp. 3464-3468). doi:10.1109/ICIP.2016.7533003

Task 2.2: Tracking with a Kalman Filter (Team 4) (2/3)

Tracking using SORT

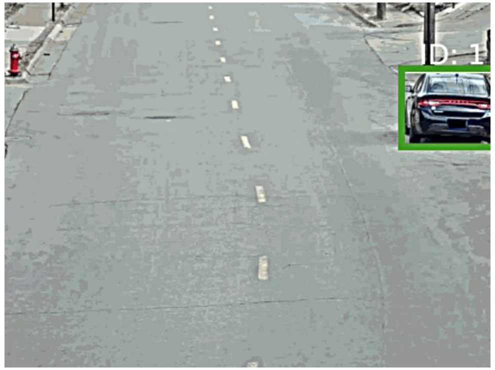


RetinaNet results using SORT. Each bounding box colour represents an ID. White bounding boxes underneath are the results of the object detection step.

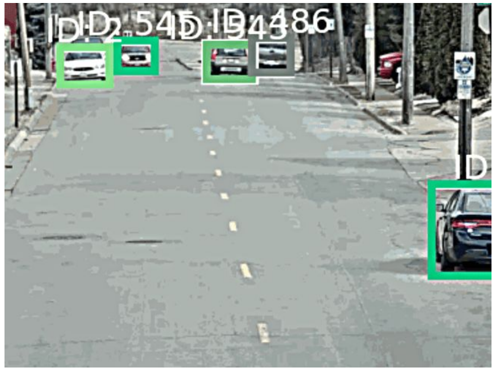
- Using SORT in not-so-good detections can decrease the score compared to other tracking algorithms, as it creates or deletes bounding boxes depending on the situation and modifies their coordinates according to the motion estimation.

We can see how now each bounding box has an ID that, in most cases, remains the same for each detection. The most troublesome area is the background, as cars are too close together and sometimes the IDs get reset or mixed up.

Some bad scenarios:



With unstable/flickery detections, SORT tends to suppress them entirely as they do not appear for enough frames in a row. Besides, IDs are not consistent as it is treated as different objects



The neural network missed one of the cars for a couple of frames after being too nearby to another car. The tracking algorithm identified it as two different IDs.

Task 2.2: Tracking with a Kalman Filter (Team 4) (3/3)

SORT tracking results over different detections

Model	**Threshold**	**IDF1**	**HOTA**
Mask-RCNN	0.5	57.563	50.602
Faster-RCNN	0.5	46.004	55.762
RetinaNet	0.5	48.833	50.403
Fine-tuned Faster-RCNN	**0.5**	**79.571**	**81.06**
Ground truth detections	-	99.519	0.994977

*Only instances classified as cars have been considered. Models are the same as the ones shown on task 2.1 (Detectron2 implementation)

Scores have been computed using TrackEval. We have obtained the best results with the Fine-tuned version of Faster-RCNN. As expected the tracker performs best when the detection is more accurate.

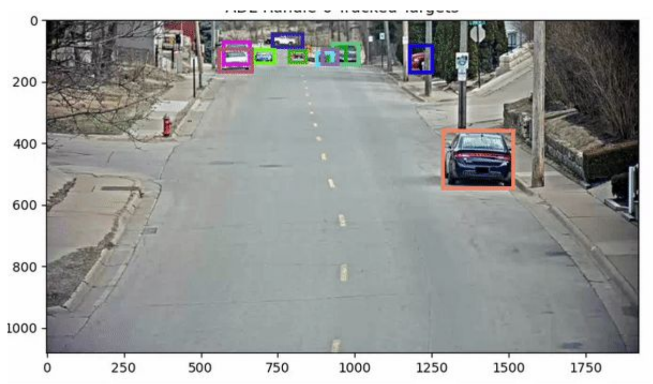
Overall, **SORT can get good results but it relies a lot on having a good detection as a starting point.**

For reference, we also ran SORT using the bounding boxes of the ground truth to see how it would perform in an ideal scenario. The scores are close to 100%, however the total detected bounding boxes after using SORT decreased by 181. It is possible that those belonged to cars on the back that did not meet the limit of frames to be considered in the tracking as they appear very briefly in the footage. Furthermore, in this case there was a higher amount of false negatives compared to false positives (194 and 13 respectively), which might be because of the algorithm creating new tracking IDs for cars that reappear after being occluded, instead of keeping their original ones as seen in the previous slide.

Task 2.2: Tracking with a Kalman Filter (Team 5)[1/3]

The TrackEval project has been used to evaluate the Kalman tracking model proposed in SORT and, since it is a robust implementation and has been a popular choice while applying the kalmar filter we have decided to go straight with the SORT project (Simple Online and Realtime Tracking).

- In the implementation of SORT, the Kalman filter is used in conjunction with a motion model to predict the position and velocity of the object in each video frame.
- The evaluation of the tracker is heavily reliant on the object detector, as its accuracy and precision have significant impacts on the overall performance, it means, working with a strong object detector will improve the results of the kalman filter.
- A pre-trained DETR model followed by fine-tuning has been proposed for this task.



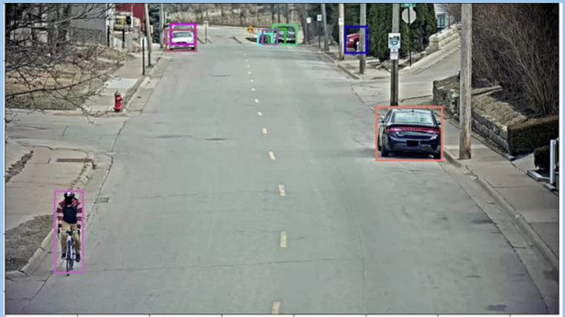
TO REPRODUCE THIS EXPERIMENT:

Object Detector: Fine tuned DETR
Video: AI_City S03_C010.AVI
GT: Full annotations .xml to .txt in MOT format
DT: MOT format in .txt
Tracker: Kalman FIlter (SORT)MOT format in .txt
Identification: By color

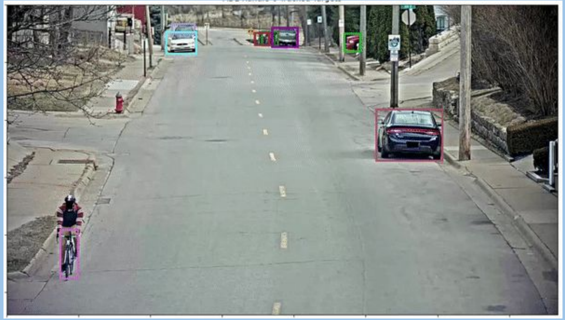
Task 2.2: Tracking with a Kalman Filter (Team 5)[2/3]

Qualitative Analysis

Fine tuned



Pretrained



It can be observed the difference of the quality of the detector / tracker tacking a look at differents characteristics of both videos, like the bounding box of the bicycle.

- In bigger objects kalman filter has no problem to handle oclussions.

- When the object detector fails to keep the detection, it has a high impact on the kalman tracker, it usually lost the track and assign another id to the object.
- Fast speed of an object and occlusion result in lost of the track.

- When objects are static in the image the kalman filter is able to keep the track of it even if there is a big occlusion.

Task 2.2: Tracking with a Kalman Filter (Team 5)[3/3]

Quantitative Analysis

DETR pre trained and fine tuned models have been proposed for this task, being the one that got better results in tasks 1.x, It can be observed in the evaluation that the fine tuning has a high impact on the results of the trackers and therefore in the evaluation. The Kalman filter results also outperforms to the overlap method, being more robust.

Fine tuned	⬜
Pretrained	⬜

Tracker/ Model	**HOTA**	**DetA**	**AssA**	**IDF1**
Maximum Overlap DETR Fine tuned	83.0	87.3	80.5	85.6
Maximum Overlap DETR Pretrained	43.4	47.9	44.0	33.9
Kalman DETR Fine tuned	86.656	86.921	87.842	93.53
Kalman DETR Pretrained	47.7	50.5	49.7	40.7

Results
fine tuned vs pretrained
task 1.3

	Percentage of improvement comparing pretrained
mAP50	**+48.3%**
mIoU	**+47.5%**

Task 2.2: Tracking with a Kalman Filter (Team 6) [1/3]

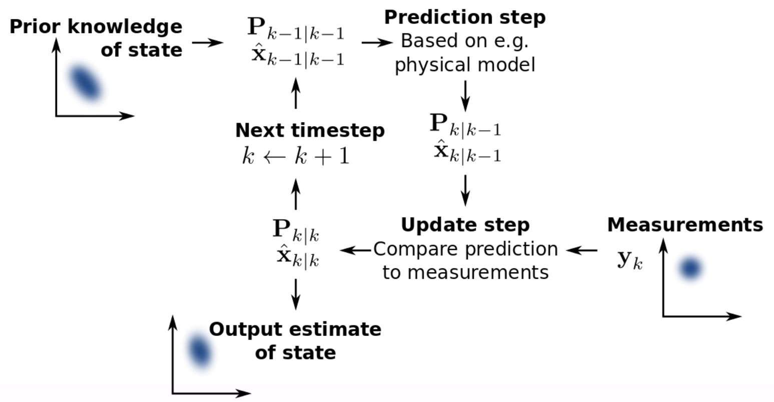
To try to improve the previous results, we used the Kalman Filter. Our implementation is based on this github implementation [1] and we consider the **constant velocity** model. Note that we are using the detections obtained from the fine tuned models (task 1.3).

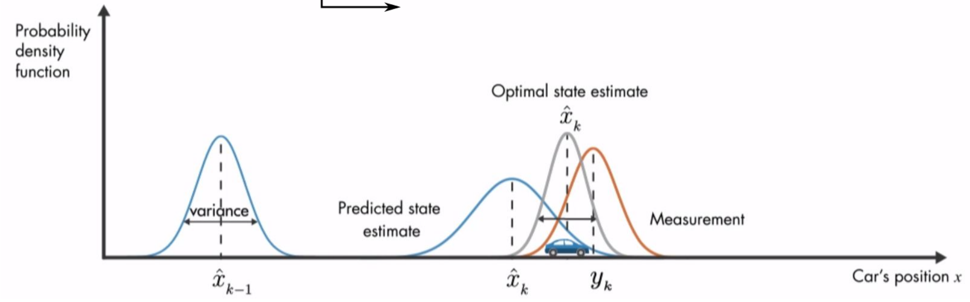
- Kalman filtering employs a sequence of measurements gathered over time, which may contain statistical noise and other inaccuracies. By estimating a joint probability distribution over the variables

- for each time frame, the algorithm produces estimates of unknown variables that are typically more precise than those based on a single measurement alone.

We will use this parameters for the filter:

1. **max_age = 10** → maximum number of frames that an object can be tracked without receiving a detection before it is considered lost and removed from the list of tracked objects.
2. **min_hits = 3** → This parameter specifies the minimum number of detections required to start tracking an object.
3. **iou_threshold = 0.3** → IoU threshold for data association, which is used to match detections to existing tracks.

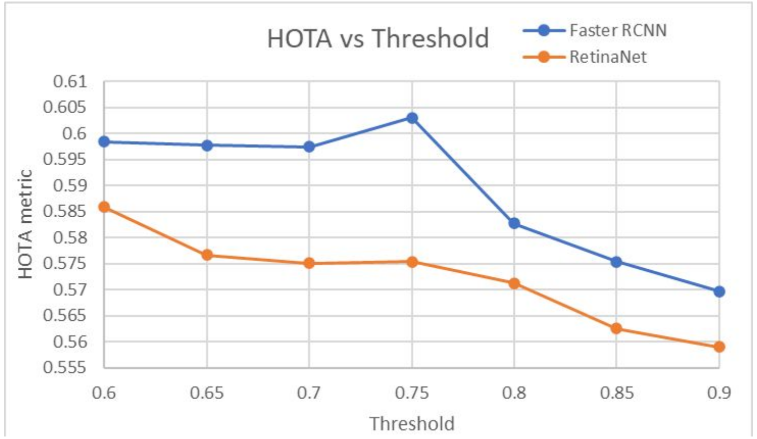

Prior knowledge of state
$\mathbf{P}_{k-1|k-1}$ $\hat{\mathbf{x}}_{k-1|k-1}$
Prediction step Based on e.g. physical model
$\mathbf{P}_{k|k-1}$ $\hat{\mathbf{x}}_{k|k-1}$
Next timestep $k \leftarrow k+1$
Update step Compare prediction to measurements
Measurements $\mathbf{y}_k$
$\mathbf{P}_{k|k}$ $\hat{\mathbf{x}}_{k|k}$
Output estimate of state

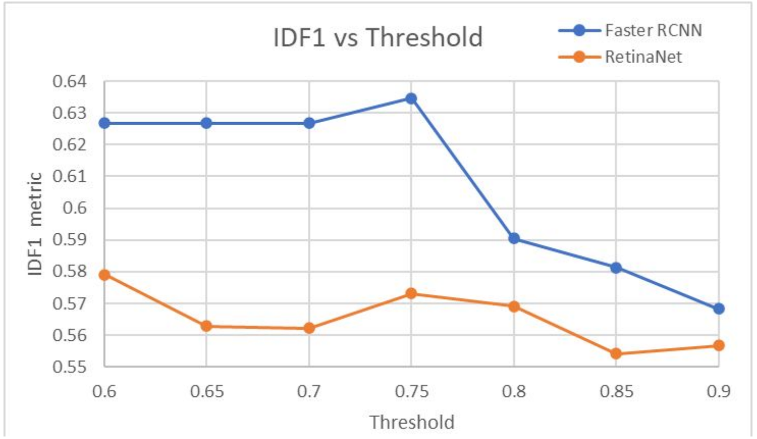


[2] [3]

Task 2.2: Tracking with a Kalman Filter (Team 6) [2/3]

- The results are obtained with the parameters mentioned in the previous slide, which we determined with some experiments. Then we did a **grid search** using different values for the **IoU threshold** of the **detections**. We are also using the filtering to not detect the bikes (explained before).
- Note that maybe we could have improved the results doing a grid over the parameters of the kalman filter, but for lack of time and the restrictions and rigidity of the repository that we were using for evaluating the tracking [1] we could not do it.





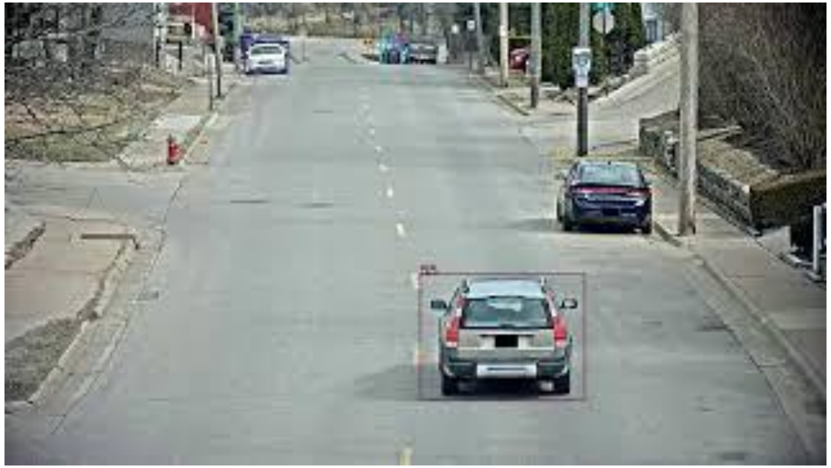
***Results with parked cars included**

- We see that using the directions of Faster RCNN results in better tracking than the ones obtained with RetinaNet. Moreover, the best threshold of the IoU for the detections seems to be 0.75.
- Note also that the results are really similar to the ones obtained with the Maximum overlap method for the IOU thr = 0.75
- In this case though, there is an abrupt change drop of performance when we increase the threshold to 0.8, so finetune seems more important here.

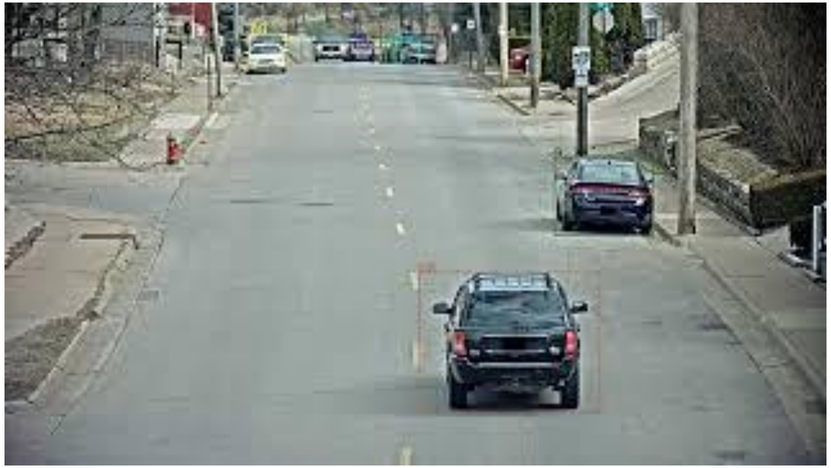
Task 2.2: Tracking with a Kalman Filter (Team 6) [3/3]

- Here we show the results with the best IoU threshold found in the grid search (0.75)

Faster RCNN



RetinaNet



- Here we see how constant velocity model properly tracks the car. Note that the initial velocity is estimated using the initial frames. Maybe, at the end of the sequence constant acceleration would have performed better since some cars in the sequence may not follow a constant velocity.
- See that we have some problems with the parked cars, since there are missing detections in the ones far away from the camera. For example, on the left cars, Faster RCNN has only one detection and RetinaNet has two intermittent detections.
- The tracking for each of the two networks are quite similar, although using the detections of Faster RCNN seems to track the cars slightly better.

Task 2.2: Feedback

	feedback
Team 1	SORT implementation No slides explaining/reviewing Kalman filtering Good explanation of hyperparameters. Good discussion of results. Performed a search for hyperparameters (be more specific: grid search, random). No IDF1/HOTA measures why?
Team 2	SORT implementation No slides explaining/reviewing Kalman filtering Good explanation of hyperparameters. Good discussion of results. Analysis on pre-trained and finetuned FRCNN.
Team 3	SORT implementation No slides explaining/reviewing Kalman filtering. Good explanation of hyperparameters. Performed a search for hyperparameters with similar results (be more specific: grid search, random) No discussion on qualitative evaluation.
Team 4	SORT implementation No slides explaining/reviewing Kalman filtering Good explanation of hyperparameters. Good discussion of results. Analysis on several models including pre-trained and finetuned ones.
Team 5	SORT implementation No slides explaining/reviewing Kalman filtering. No explanation of hyperparameters. Bonus points for a very good presentation of results and discussions.
Team 6	SORT implementation Slides explaining/reviewing Kalman filtering Good explanation of hyperparameters. Good discussion of results.

Tasks

- Task 1: Object detection
- Task 2: Object tracking
    - Task 2.1: Tracking by overlap
    - Task 2.2: Tracking with a Kalman Filter
    - **Task 2.3: IDF1,HOTA scores**

Task 2.3: IDF1 for Multiple Object Tracking

Team ID	T2.1 Tracking by overlap		T2.2 Tracking with a Kalman filter	
	IDF1	HOTA	IDF1	HOTA
Team 1	81.392	78.97	68.586	72.771
Team 2	69.30	63.66	78.42	74.27
Team 3	80.40		84.60	
Team 4	76.938	80.878	79.571	81.06
Team 5	85.7	83.0	93.53	86.656
Team 6	0.634	0.605	0.635	0.603

Task 2.3: Feedback

	feedback
Team 1	
Team 2	
Team 3	no hota. why?
Team 4	
Team 5	
Team 6	

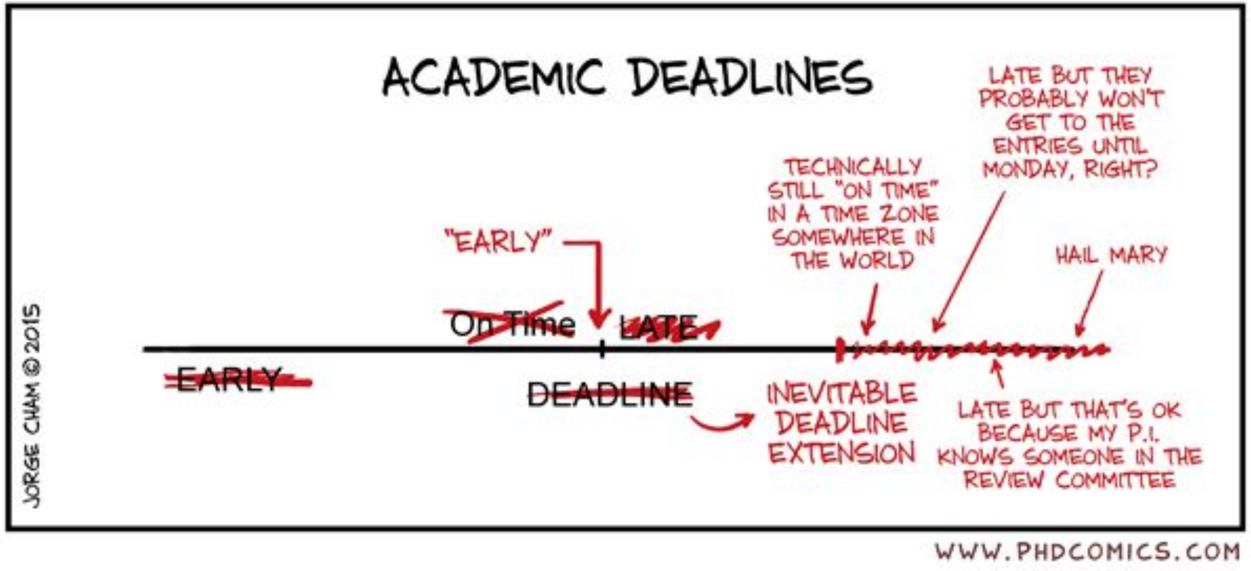
Tasks

- Task 1: Object detection
- Task 2: Object tracking
- **(optional) Task 3: CVPR 2021 AI City Challenge**

Scoring Rubric

Task	Description	Max, Score
T1.1	Off-the-shelf	1
T1.2	Annotate	1
T1.3	Fine-tuning	2
T1.4	K-fold Cross validation	1
T2.1	Tracking by Overlap	2
T2.2	Tracking with a Kalman Filter	2
T2.3	IDF1 for Multiple Object Tracking	1
T3	(optional) CVPR 2021 AI City Challenge	+1

Deliverables


ACADEMIC DEADLINES
~~EARLY~~
~~On Time~~
"EARLY"
~~LATE~~
~~DEADLINE~~
INEVITABLE DEADLINE EXTENSION
TECHNICALLY STILL "ON TIME" IN A TIME ZONE SOMEWHERE IN THE WORLD
LATE BUT THEY PROBABLY WON'T GET TO THE ENTRIES UNTIL MONDAY, RIGHT?
HAIL MARY
LATE BUT THAT'S OK BECAUSE MY P.I. KNOWS SOMEONE IN THE REVIEW COMMITTEE
JORGE CHAM © 2015
WWW.PHDCOMICS.COM

- Deadline: **Wednesday March 29th at 3pm**
- Deliverables:
  - Submit your report by editing these slides: task 1 and task2
  - Provide feedback regarding the teamwork (email)