Master in
Computer Vision
Barcelona

Project
Module 6
Coordination

Video Surveillance for Road Traffic Monitoring

J. Ruiz-Hidalgo / X. Giró

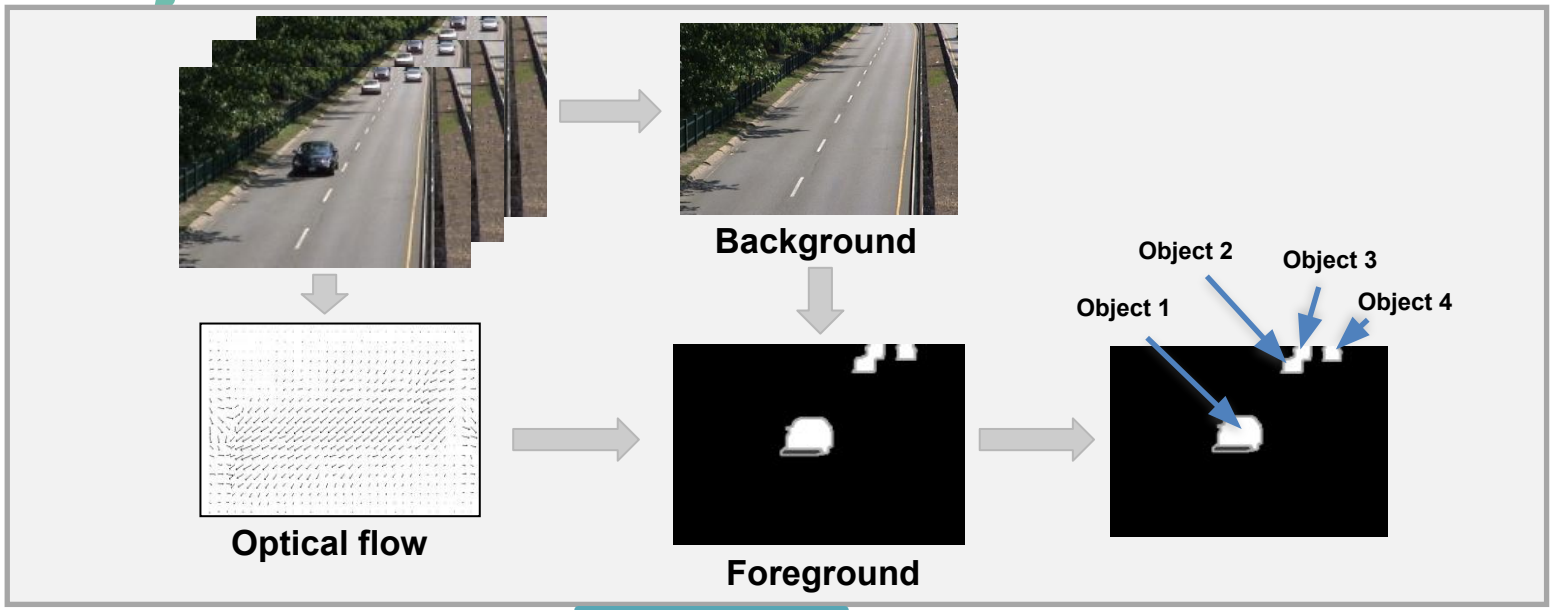
j.ruiz@upc.edu / xavier.giro@upc.edu

Week 3: Instructions

Master in
Computer Vision
Barcelona

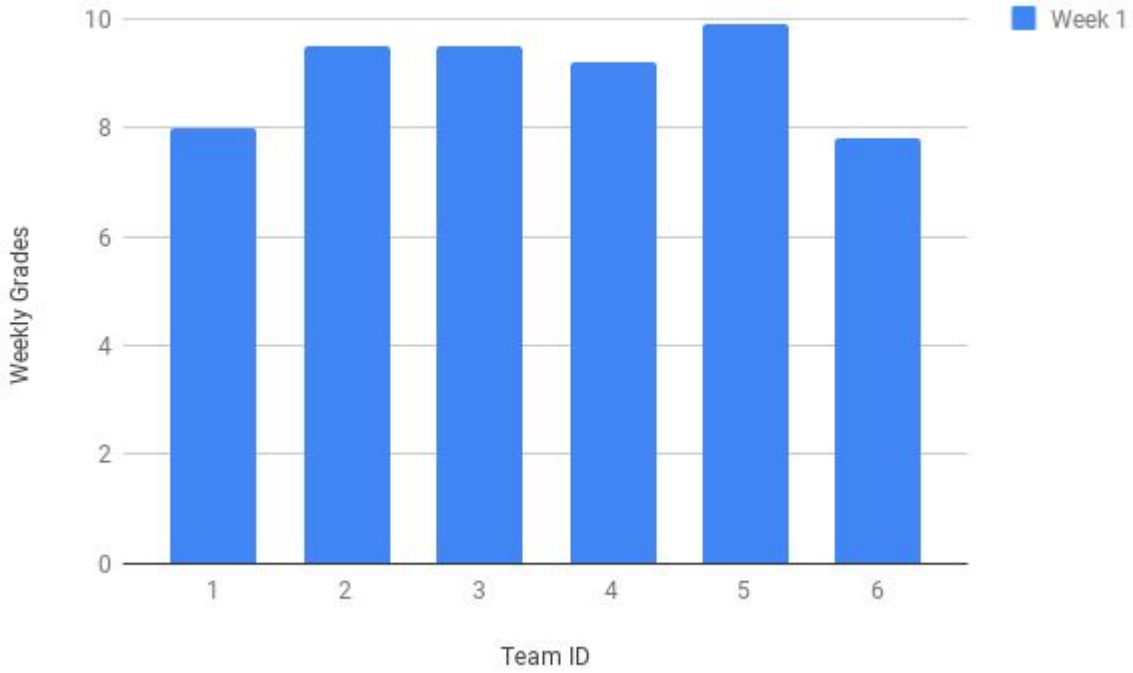
UAB UOC UPC upf.

Project Schedule



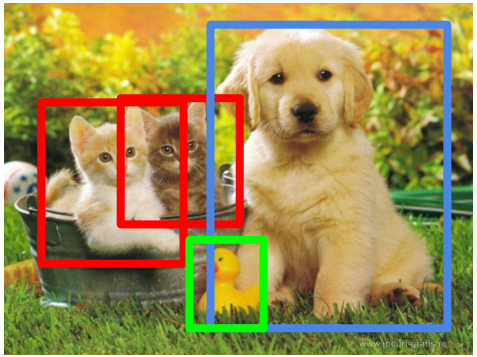
Week 1	Week 2	Week 3	Week 4	Week 5	Week 6
• Introduction • DB • Evaluation metrics	• Background estimation • Stauffer & Grimson	• Object Detection • Tracking	• Optical flow • Tracking	• Multiple cameras • Speed	• Presentation workshop

Average grades per team



Goals Week 3

Object detection



CAT, DOG, DUCK

The task of assigning a **label** and a **bounding box** to all objects in the image

Object tracking

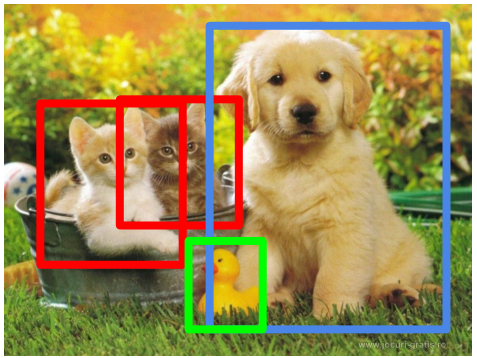
Sequence S03 - C010

Tasks

- Task 1: Object detection
- Task 2: Object tracking

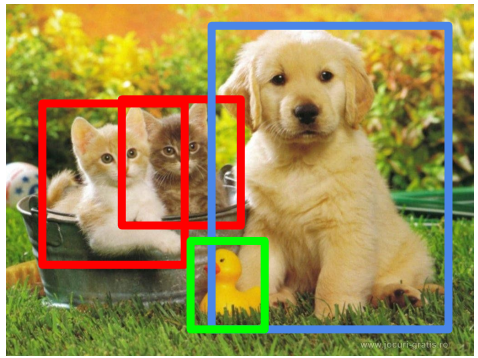
Tasks

- **Task 1: Object detection**
- Task 2: Object tracking



Tasks

- Task 1: Object detection
  - **Task 1.1: Off-the-shelf**
  - Task 1.2: Fine-tune to your data
- Task 2: Object tracking



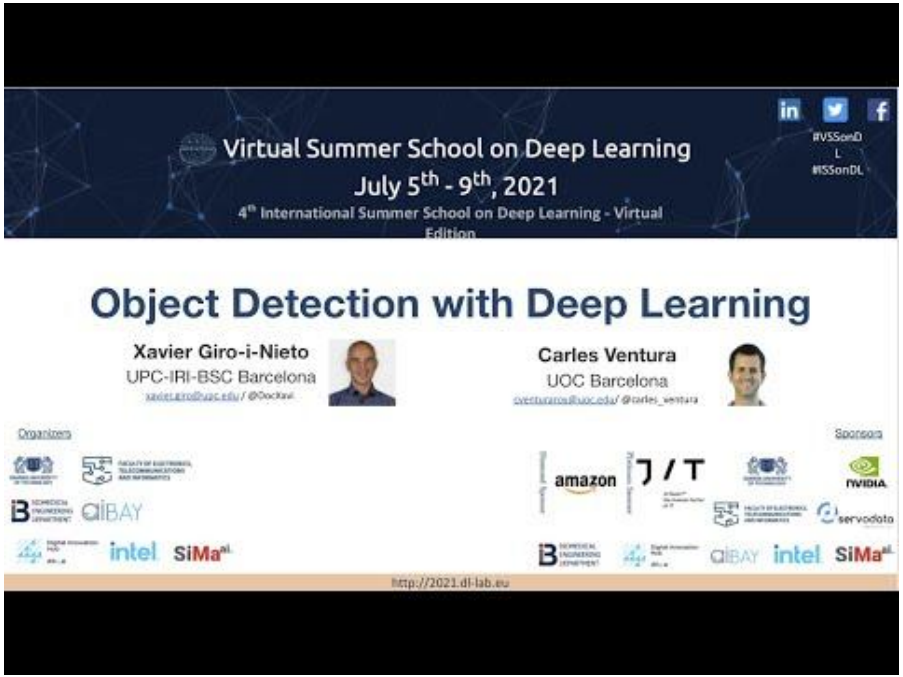
Task 1: Object Detection - Video Lectures

M5 Lectures by

LLuís Gómez

Video-lecture from

Xavier Giró (UPC) & Carles Ventura (UOC) [Slides] [More]



Task 1: Object Detection

More slides & Videos from UPC TelecomBCN (all) & UPC School

Object Detection

Xavier Giro-i-Nieto	AIDL2020	Slides	
Andreu Girbau	DLCV2019		Video
Miriam Bellver	DLCV2018	Slides	Video
Amaia Salvador	DLCV2017	Slides	Video
Amaia Salvador	DLCV2016	Slides	Video

Task 1.1: Object Detection: Off-the-Shelf

Use an existing one or more object detectors to detect cars in each frame (separately). Examples:

Model	Framework
Benchmark, Mask R-CNN	TensorFlow Model Garden
Faster R-CNN, Mask R-CNN	**Torchvision**
Faster R-CNN, RetinaNet…	**[Detectron2]**
DETR	[PyTorch]
YOLOv3	[Ultralytics]
SSD	[PyTorch] [Tutorial on Keras]
Latest	YOLOX, ViDT (ICLR 2022)

Task 1.1: Object Detection: Off-the-Shelf

Model Name	Model size	Device	GPU	CPU
COCO SSD MobileNet v1	27 Mb	Pixel 3 (Android 10)	22ms	46ms*
		Pixel 4 (Android 10)	20ms	29ms*
		iPhone XS (iOS 12.4.1)	7.6ms	11ms**

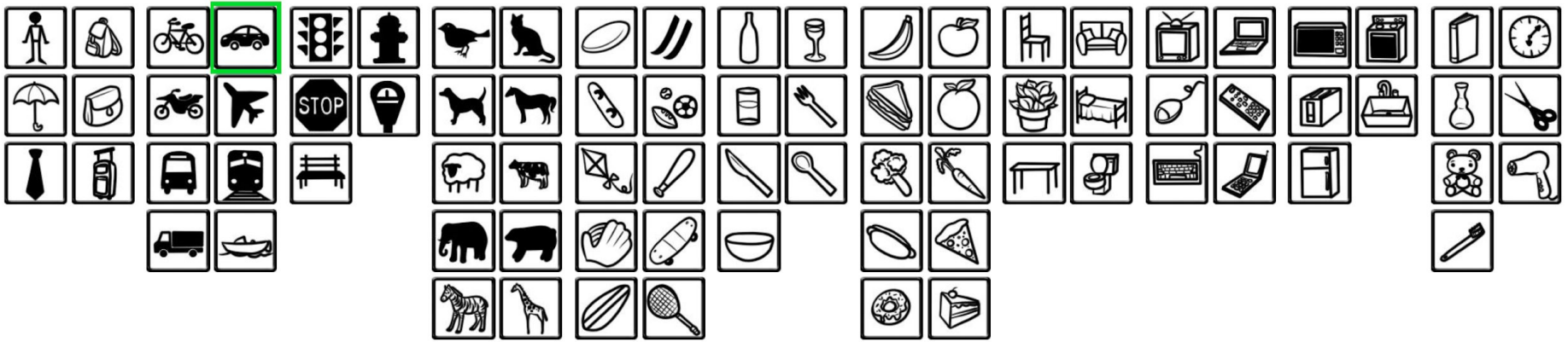
TensorFlow Lite: Object Detection
PyTorch Mobile (D2GO for object detection)

Task 1.1: Object Detection: Off-the-Shelf

Make sure that the dataset used to train your network contains the class ‘car’.

Microsoft CoCo

‘car’ is Class ID =3
[labels] [tutorial]



Task 1.1: Object Detection: Off-the-Shelf

You actually already have the results of Mask-RCNN, SSD & YOLOv3 in the the provided data.

We do not know which implementation was used.

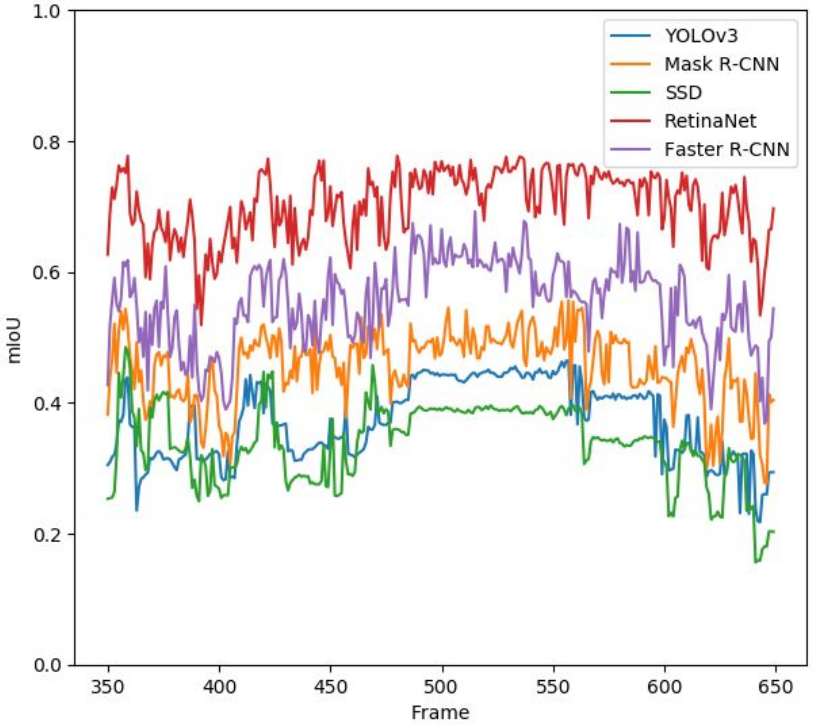
Figure: Team 5 (2018/2019)

Task 1.1: Object Detection: Off-the-Shelf

Slide: Team 4 (2019/2020)

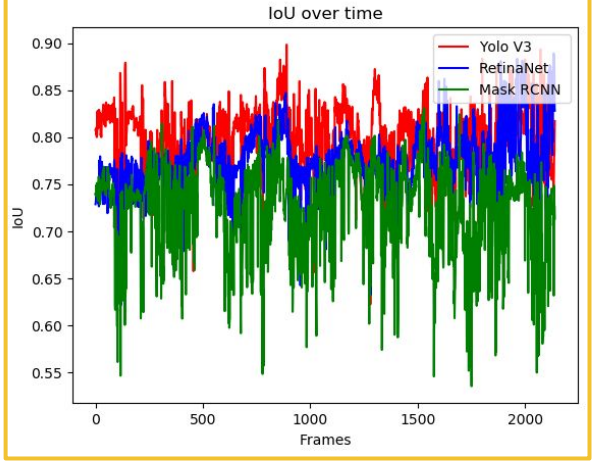
We can observe below a plot of our results alongside the provided detections. However, we believe that this is not a fair comparison because we don't know the source of the provided detections and it could have very different training conditions.

	$AP_{0.5car}$
RetinaNet (R50-FPN)	0.436
Faster R-CNN (R50-FPN)	**0.595**
Mask R-CNN	0.447
SSD 512	0.382
Yolo v3	0.435



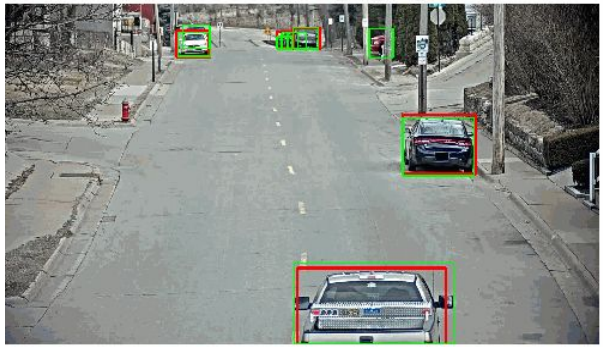
Task 1.1: Object Detection: Off-the-Shelf

Slide: Team 6 (2019/2020)

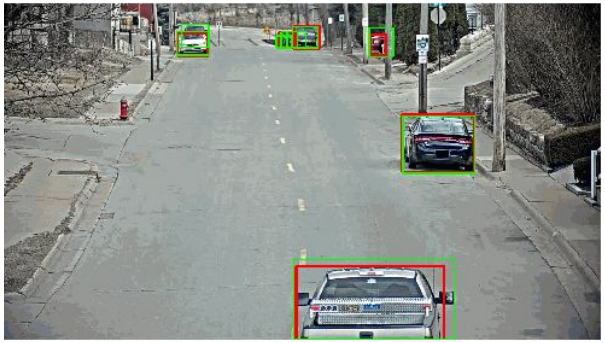


Qualitative results:

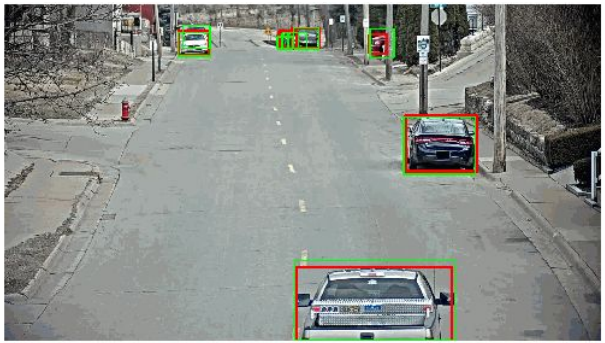
YOLO V3



RetinaNet



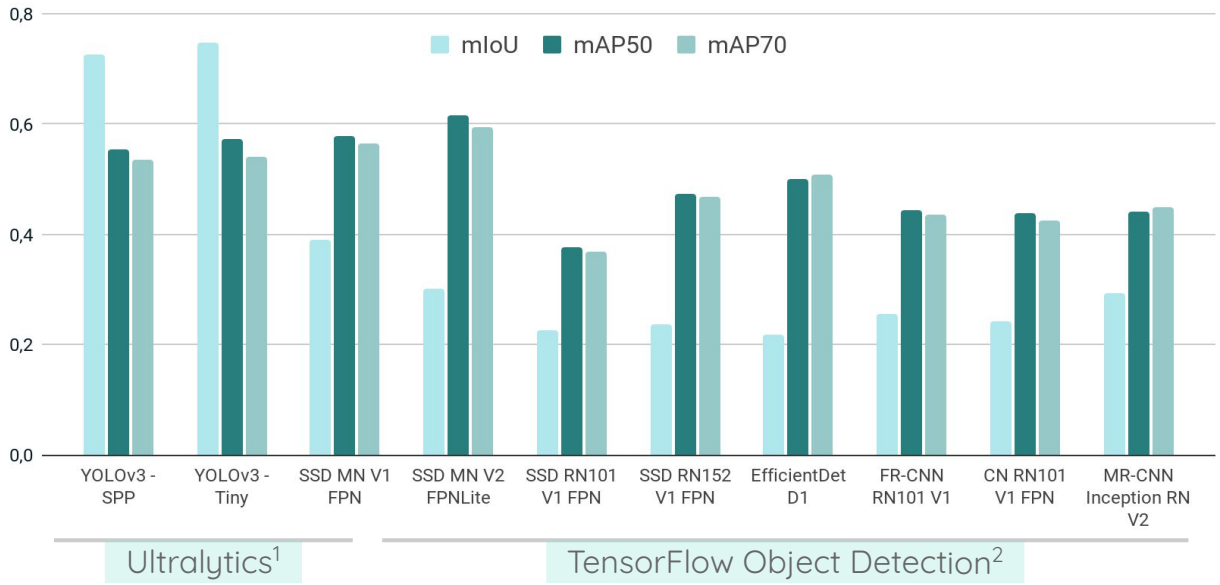
Mask RCNN



Detections
Groundtruth

Task 1.1: Object Detection: Off-the-Shelf

Slide: Team 3 (2021)



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

It exists a significant difference in the IoU metrics between the YOLO Ultralytics and the TensorFlow ones. This is mainly because the bounding boxes estimated by TensorFlow models are **noisier** than the ones from YOLO, making their **overlap smaller**.

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

Task 1.1: Object Detection: Off-the-Shelf

Slide: Team 4 (2021)

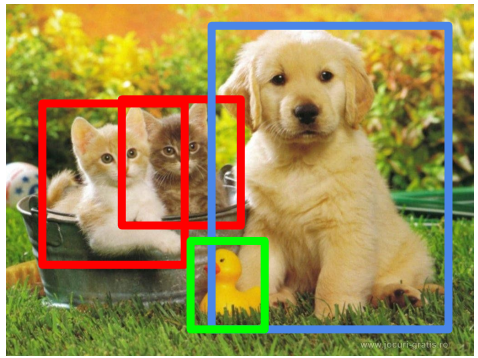
Model	Implementation	Confidence threshold	$AP^{0.5}$	$AP^{0.7}$	Inference time (s/img)
Faster R-CNN	**Detectron2 (R_50_FPN_3x)**	**0.5**	**0.602**	0.433	0.1049
		0.7	0.532	0.433	
RetinaNet	Detectron2 (R_50_FPN_3x)	0.5	0.531	0.453	0.09886
		0.7	0.364	0.364	
YOLOv3	Pytorch	0.5	0.543	0.442	0.03099
		0.7	0.543	0.442	
Mask R-CNN	Keras	0.5	0.483	0.399	0.51599
		0.7	0.483	0.399	

Task 1.1: Object Detection: Off-the-Shelf (Team X)

Copy & paste (max 1 page)

Tasks

- Task 1: Object detection
  - Task 1.1: Off-the-shelf
  - **Task 1.2: Fine-tune to your data**
- Task 2: Object tracking



Task 1.2: Fine-tune to your data

Lectures

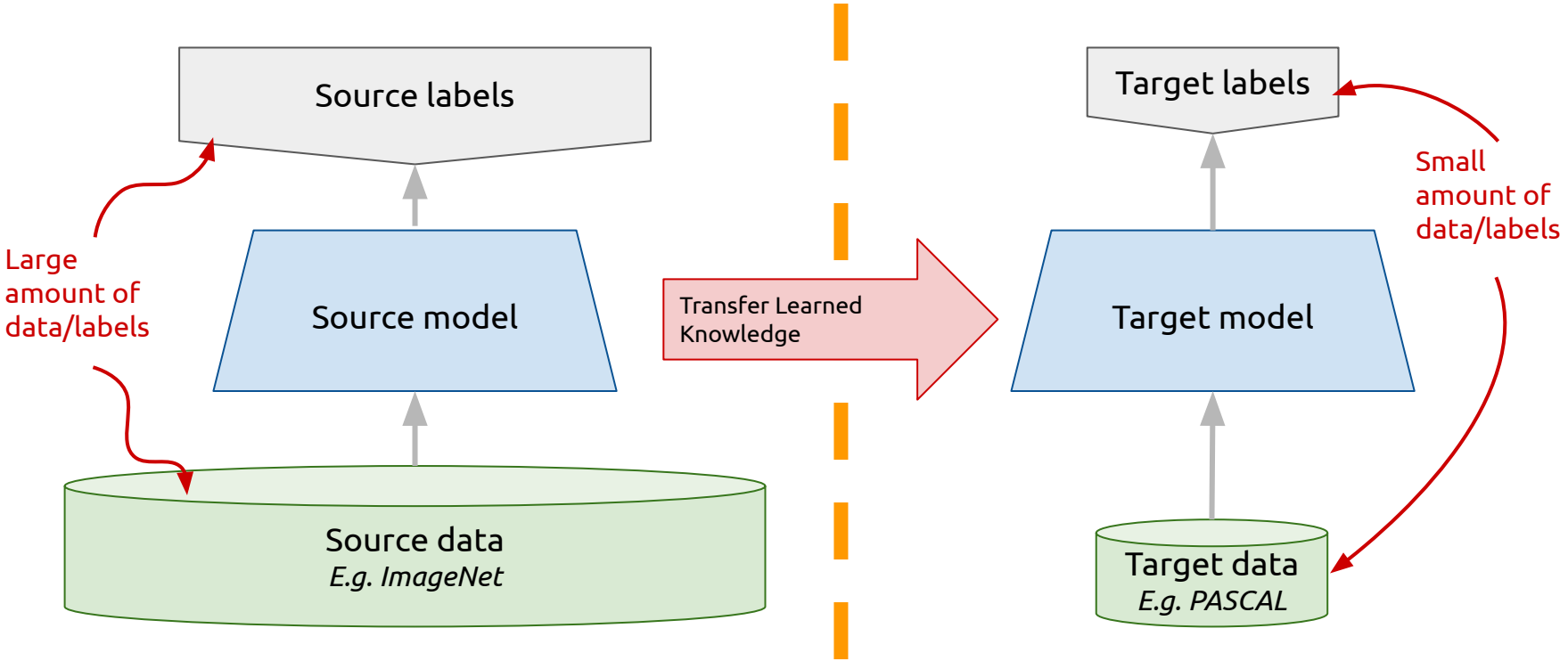
[Kevin McGuinness](#)
([UPC DLCV 2016](#))

[Ramon Morros](#)
([UPC DLAI 2018](#))

Lab

[“Transfer Learning”](#)
Míriam Bellver (2018)

Task 1.2: Fine-tune to your data



Slide credit: Kevin McGuinness (UPC DLCV 2016)

Task 1.2: Fine-tune to your data

Fine tuning from your data will in general require two steps:

- Defining the new dataset
  - (eg. here for Faster R-CNN)
- Fine-tuning the last layer(s) from a pre-trained model
  (eg. here for Faster R-CNN)

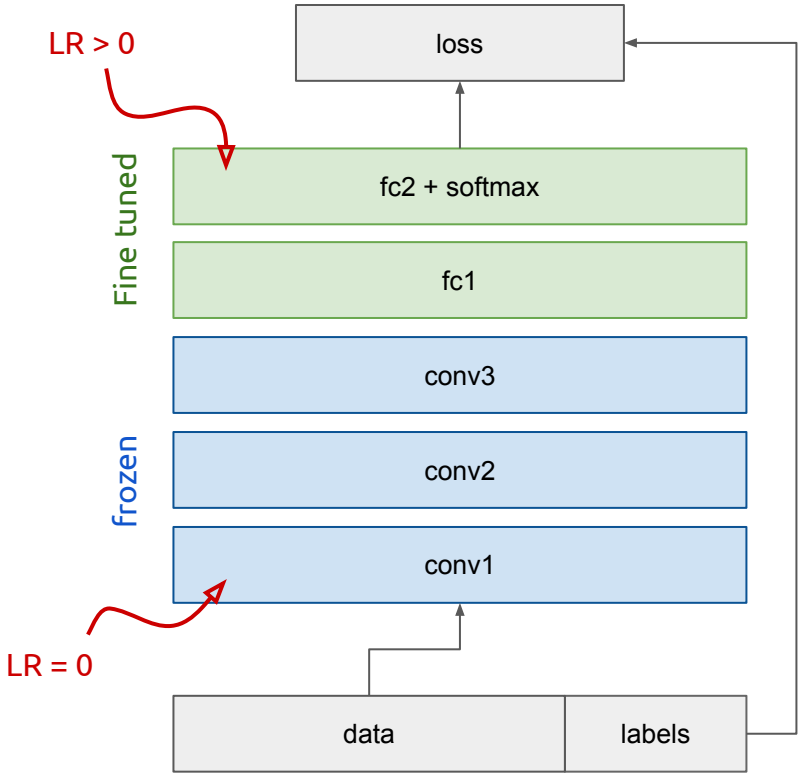
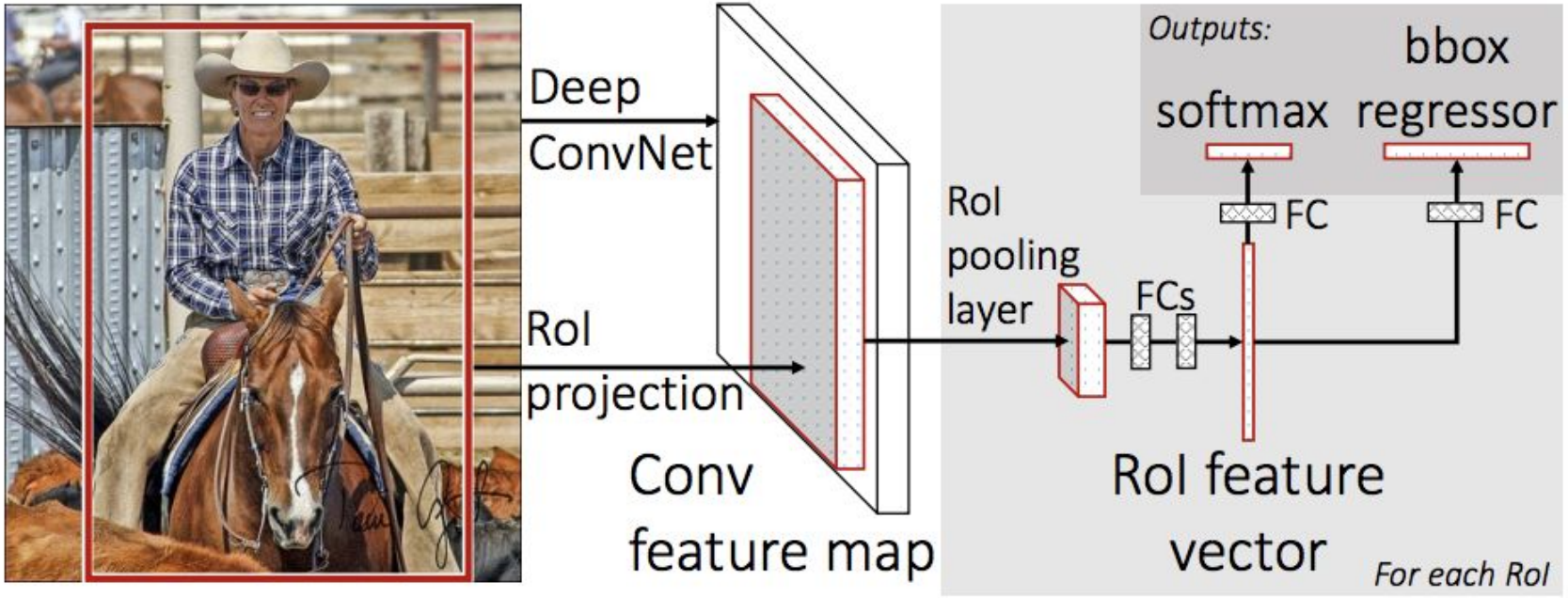


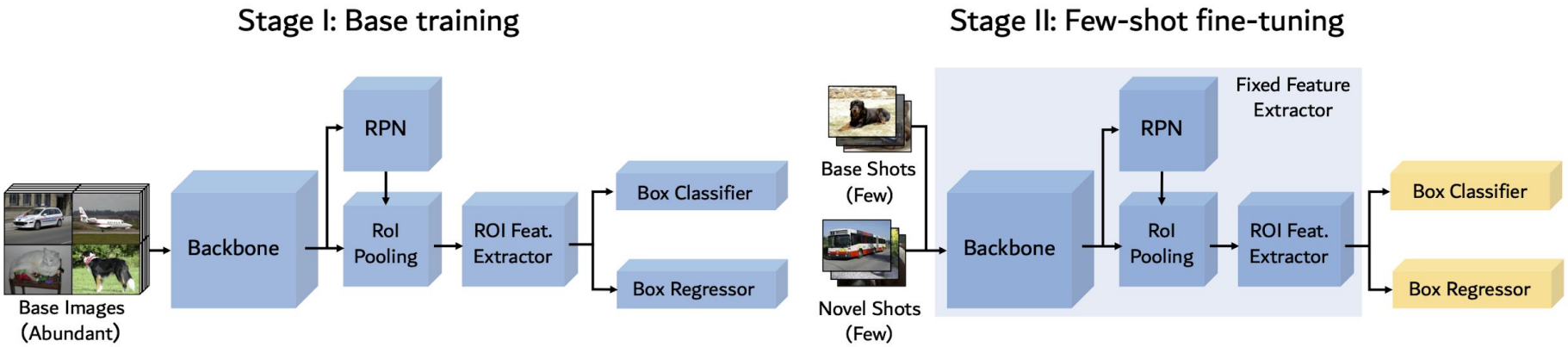
Figure credit: Kevin McGuinness (UPC DLCV 2016)

Task 1.2: Fine-tune to your data

[Tutorial from Torchvision for Mask R-CNN](#) (you do not have object segmentations)



Task 1.2: Fine-tune to your data

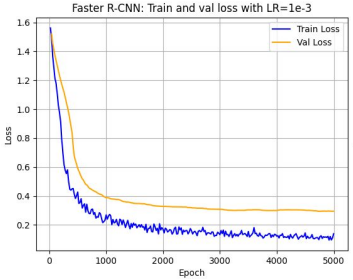


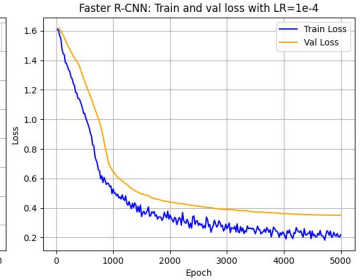
Wang, Xin, Thomas E. Huang, Trevor Darrell, Joseph E. Gonzalez, and Fisher Yu. "Frustratingly Simple Few-Shot Object Detection." arXiv preprint arXiv:2003.06957 (2020). [code based on Detectron 2]

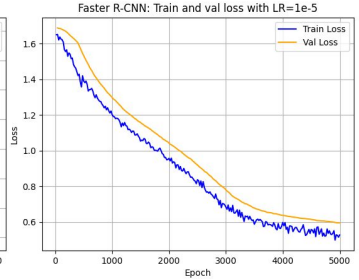
Task 1.2: Fine-tune to your data

Slide: Team 4 (2021)

Model	LR	Validation $AP^{0.5}$
Faster R-CNN	**1e-3**	**0.969**
	1e-4	0.968
	1e-5	0.946







Model	Freeze Stages	Validation $AP^{0.5}$
Faster R-CNN	0	0.970
	1	0.969
	2	0.969
	3	0.969
	4	0.968

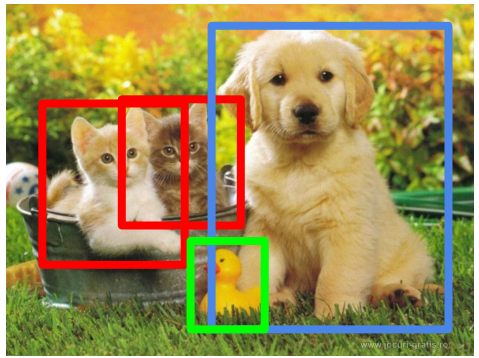
	Augmentation: Random			
Model	**Horizontal Flip (prob 0.5)**	**Brightness (0.9, 1.1)**	**Crop (640, 640)**	**Validation $AP^{0.5}$**
Faster R-CNN	X			0.823
		X		0.819
			X	0.781
	X	X	X	0.841

Task 1.2: Object Detection: Fine-tune (Team X)

- Copy & paste (max 2. pages)

Tasks

- Task 1: Object detection
  - Task 1.1: Off-the-shelf
  - Task 1.2: Fine-tune to your data
  - **Task 1.3: K-Fold Cross-validation**
- Task 2: Object tracking



Task 1.3: K-Fold Cross-validation

Tutorial and example: [Cross-validation tutorial in scikit-learn](#).

Task 1.3: K-Fold Cross-validation

Try different data partitions on your sequence:

Strategy A (same as week 2):

- First 25% frames for training
- Second 75% for test.

Strategy B:

- K-Fold cross-validation (use K=4).
- First 25% Train - last 75% Test (same as Strategy A).

Strategy C:

- K-Fold cross-validation (use K=4)
- Random 25% Train - rest for Test

Task 1.3: K-Fold Cross-validation

Slide: Team 4 (2021)

			Validation $AP^{0.5}$ for split #			
Model	**Strategy**	**LR**	**0**	**1**	**2**	**Mean $AP^{0.5}$**
Faster R-CNN	B	1e-3	0.99	0.92	0.97	0.96
		1e-4	0.99	0.89	0.97	0.95
		1e-5	0.95	0.87	0.94	0.92
	C	1e-3	0.98	0.99	0.98	0.98
		1e-4	0.96	0.97	0.96	0.96
		1e-5	0.90	0.91	0.89	0.90

Faster R-CNN fine tuned using strategy	Test $AP^{0.5}$
A	0.966
B	0.966
C	0.98

Task 1.3: K-Fold Cross-validation (Team X)

- Copy & paste (max 1. pages)

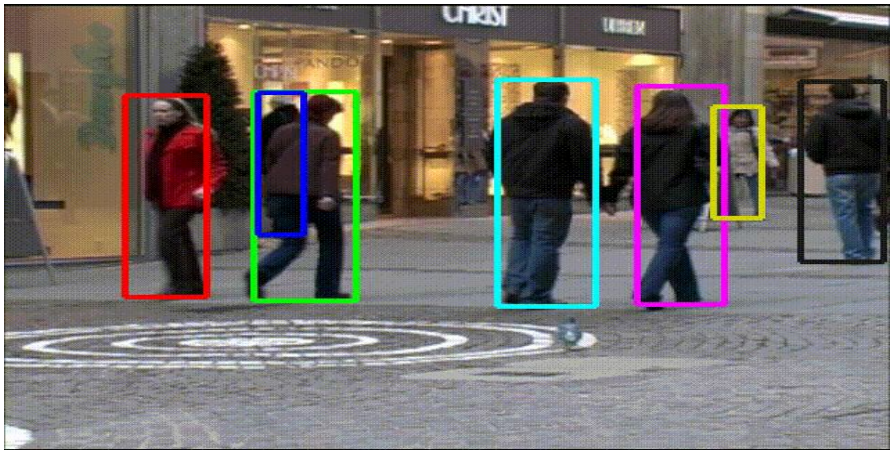
Task 1: Object Detection

Team ID	Model (choose one)	T1.1 Off-the-shelf (mAP_{50}) (Random 3-Fold cross-validation)	T1.2 Fine-tuned (mAP_{50}) (Random 3-Fold cross-validation)
Team 1			
Team 2			
Team 3			
Team 4			
Team 5			
Team 6			

Tasks

- Task 1: Object detection
- **Task 2: Object tracking**

T2. Object Tracking



Challenge:
Assign a **unique ID** to objects across a video.

Set up:
The object initial location is known.

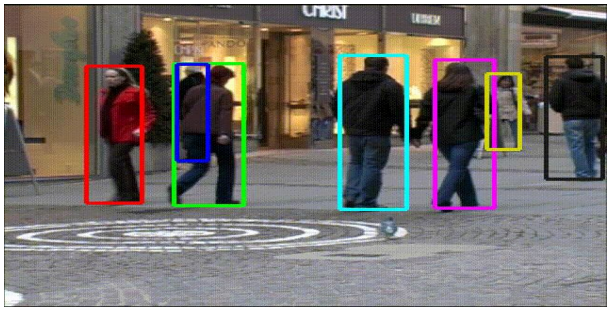
T2. Object Tracking

Single object tracking (SOT)

Usain Bolt

gif source

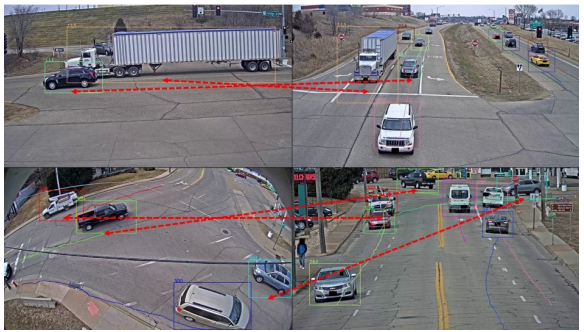
Multiple Object Tracking (MOT)



(P1, P2, P3, P4 ...)

video source

Multi-Target Multi-Camera Tracking (MTMC)



Source:
AI City Challenge 2019

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 2.1: Tracking by Maximum Overlap

Video-lecture from

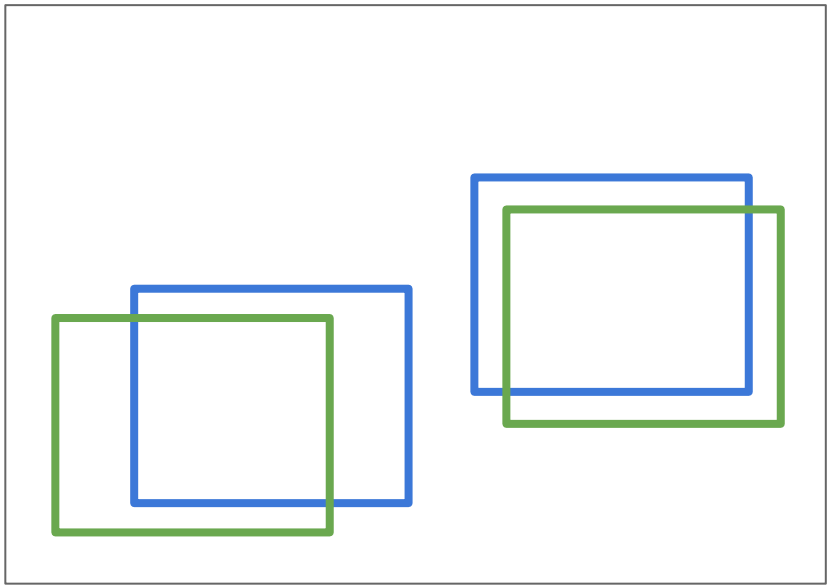
UPC TelecomBCN Summer School

Laura Leal-Taixé (2018)

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

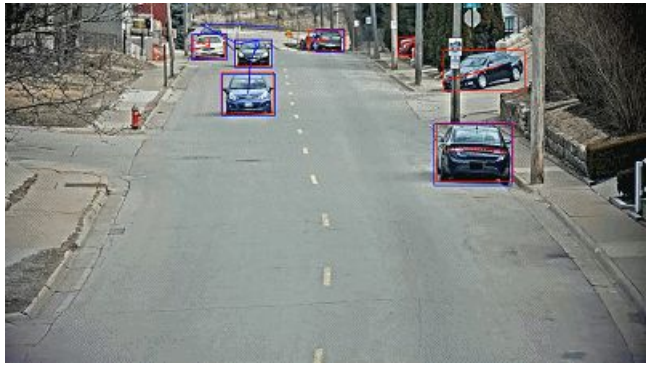
Detected Objects @ Frame N+1

Task 2.1: Tracking by Maximum Overlap

Slide: Team 3 (2019/2020)

First, we use the basic algorithm to implement the tracking.

1. Define a new track for every new detected object in frame N.
2. In frame N+1, assign the same ID to the detected object with the highest overlap (IoU).
3. Return to 1.

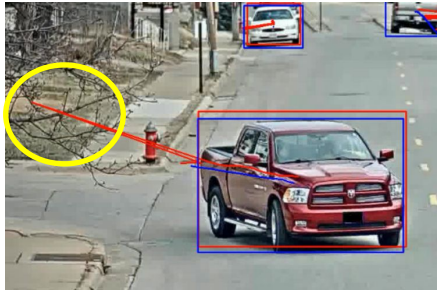


Result of basic algorithm

The result looks not bad. There are **2 main errors** that affect the accuracy of the result.

1. A long tracking may be split because of the missing detection in a few frames. (In error 1, the long tracking is cut off in the middle.)
2. The noisy detections may introduce many noisy tracking lines. (In error 2, it is affected by the wrong detection on the tree.)

Error 1



Error 2

Task 2.1: Tracking by Maximum Overlap

Slide: Team 4 (2020)

Pipeline

The results we show in this section are for the provided detections of Mask R-CNN

Pre-processing

Filter for overlapping bounding boxes with IoU larger than 0.9 to remove repeated masks.

Before

After

One of the purple bounding boxes is removed.

First algorithm

Compare bounding boxes from frame N to frame N+1. There are 2 scenarios:

1. First frame, we assign a different track to each object on the scene.
2. Following frames, each bounding box in the frame N+1 is compared to the bounding boxes in frame N. There are two different cases:
   a. If a bounding box was already labeled in the frame N, the same track value has to be assigned to the matching bounding box in the frame N+1. For matching, instead of just taking the maximum, we set a threshold of IoU ≥ 0.4 to determine the correspondence of consecutive bounding boxes.
   b. If a new bounding box in the frame N+1 has no correspondence N then a new track label is assigned to it.

Task 2.1: Tracking by Maximum Overlap

Slide: Team 1 (2021)

Quantitative results:

*

Detector	IDF1 %	IDP %	IDR %	Recall %	Precision %
RetinaNet-50	31.1	93.3	67.0	89.3	27.0
RetinaNet-101	30.2	75.2	56.8	83.6	30.2
Mask-RCNN	**42.7**	69.2	41.3	48.2	51.4
SSD	41.9	76.6	29.7	36.9	88.9
YOLO	30.9	50.7	25.2	42.4	67.4

Retina characterized by high recall and low precision (lots of FP)

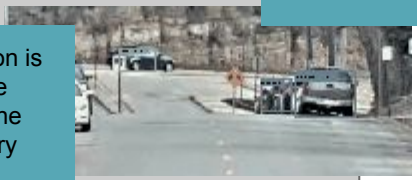
More balanced precision/recall (similar number of FP and FN)

SSD and YOLO higher precision than recall (lots of FN)

Qualitative results:

This is expected to improve with Kalman

For **farthest cars**, when the detection is lost (even for a couple of frames) the **tracking fails**. That is because for the **IoU** becomes very **low** or zero in very small time.



The **detection** (in yellow) **glitches** (probably because is more a van than a car). When the detection is only lost for a couple of frames the ID remains the same, but after a **longer miss, the ID changes**

Metrics were computed using **py-motmetrics** - Metrics for multiple object tracker (MOT) benchmarking.
IDF1: ratio of correctly identified detections over the average number of ground-truth and computed detections.
IDP: identification precision.
IDR: identification recall. Fraction of computed (ground truth) detections that are correctly identified
Precision: Number of detected objects over sum of detected and false positives.
Recall: Number of detections over number of objects.

Task 2.1: Tracking by Maximum Overlap (Team X)

- Copy & paste (max 2. pages)

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

You may get inspiration from this tutorial by Amaia Salvador to track the detections with a Kalman filter.

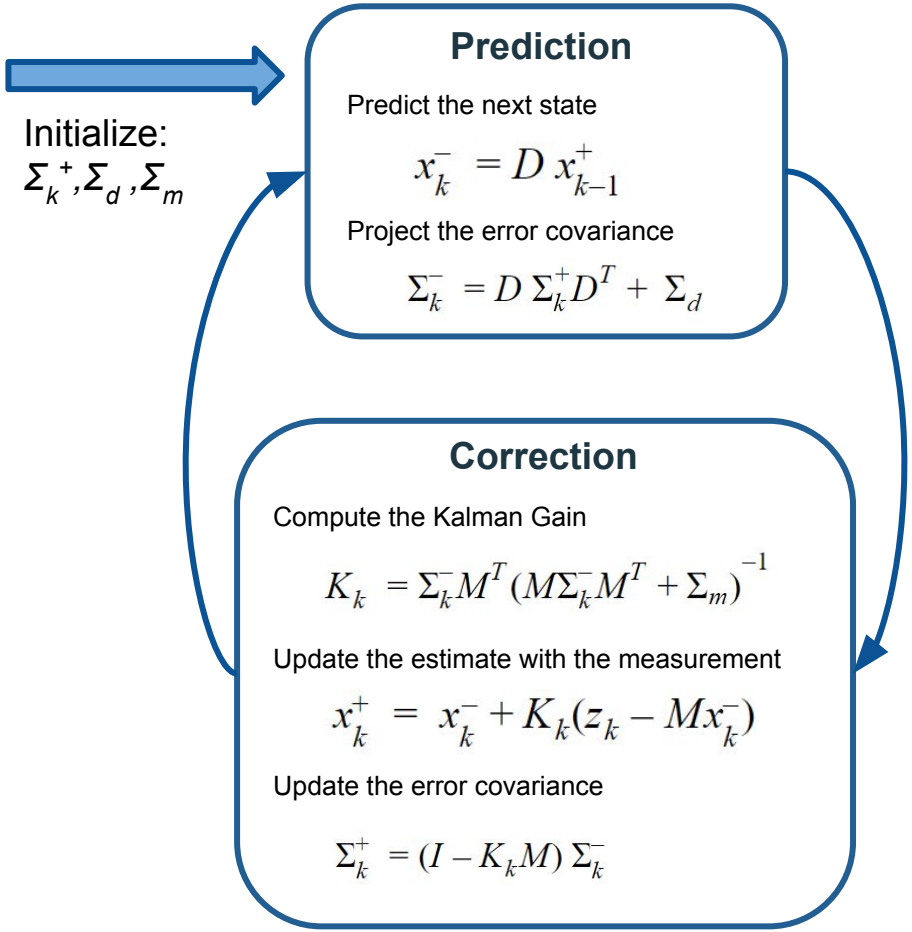
Amaia Salvador

PhD 2019

Universitat Politècnica de Catalunya

Task 2.2: Tracking with a Kalman Filter

Team 4 (2018/2019)



2 different dynamic models implemented:

- Constant velocity model

$$x_k = [x \quad y \quad \dot{x} \quad \dot{y}]_k$$

$$D = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

- Constant Acceleration model

$$x_k = [x \quad y \quad \dot{x} \quad \dot{y} \quad \ddot{x} \quad \ddot{y}]_k$$

$$D = \begin{bmatrix} 1 & 0 & \Delta t & 0 & \frac{1}{2}\Delta t^2 & 0 \\ 0 & 1 & 0 & \Delta t & 0 & \frac{1}{2}\Delta t^2 \\ 0 & 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad M = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Task 2.2: Tracking with a Kalman Filter

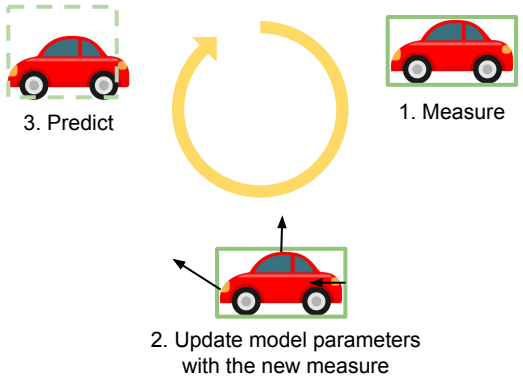
Team 1 (2021)

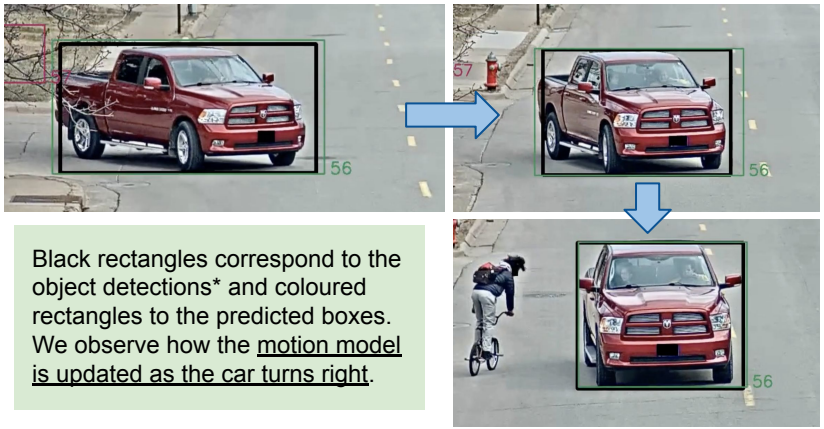
We improved our intersection-over-union tracker by including a motion model to predict the future object detections (Kalman filters). We implemented the linear constant velocity model proposed in SORT[1], where each target is modeled as a vector ***x* = [*u*, *v*, *r*, *u'*, *v'*, *s'*]**, where:

- *u*, *v* — horizontal and vertical pixel locations of the target center, respectively.
- *s*, *r* — the area and the aspect ratio of the target's bounding box, respectively.
- *u'*, *v'*, *s'* — velocities of *u*, *v* and *s*, respectively

The aspect ratio, *r*, is considered an intrinsic time lasting property of the object so it is considered to be constant.

Kalman filters loop





Black rectangles correspond to the object detections* and coloured rectangles to the predicted boxes. We observe how the motion model is updated as the car turns right.

[1] Bewley, Alex et al. "Simple Online and Realtime Tracking." 2016 IEEE International Conference on Image Processing (ICIP) (2016).

* Detections retrieved with pre-trained RetinaNet

Task 2.2: Tracking with a Kalman Filter

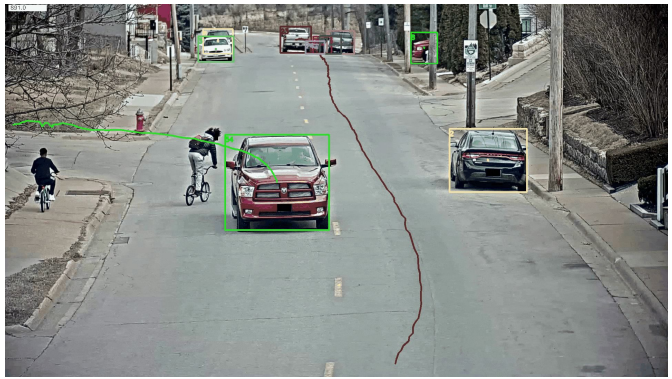
Team 4 (2021)

Our first idea was to implement the kalman filter from scratch, so we tried to develop this implementation adding different functionalities. At the end, due to time limitations, we thought it would be more interesting to directly use SORT and study the importance of its parameters.

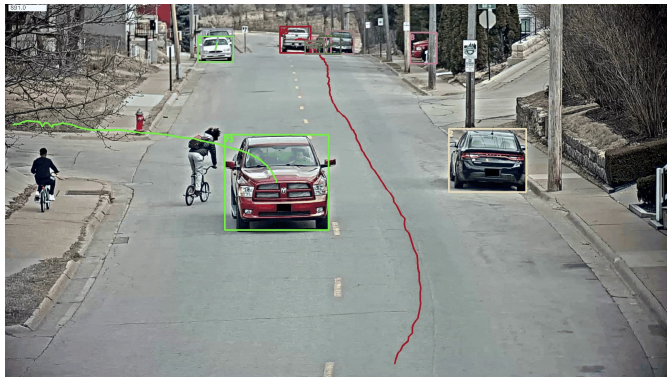
The main advantage using a Kalman filter is the robustness over missing data on frames. It allows to predict the position of the object tracked, making an approximation of the position where a bounding box from a previous frame would be located on the frame which said id is missing. This is one of the problems we faced with the maximum overlap method.

We also included the parked algorithm on the Kalman filter. This algorithm helped much more the Kalman raw implementation (+ 3%) than in the solution using the SORT algorithm (+ 1%), as the latter does cover this issue in its logic.

	IDF1 (Fine-tuned Faster RCNN)
SORT	0.837
SORT with parked algorithm	0.848
Raw Kalman	0.803
Raw Kalman with parked algorithm	0.832



SORT Kalman



Raw Kalman

Task 2.2: Tracking with a Kalman Filter (Team X)

- Copy & paste (max 2. pages)

Task 2.3: IDF1 for Multiple Object Tracking

There exist different metrics for Multiple Object Tracking that take into account the whole trajectory of the object. We focus on

- **Truth-to-result match:** a bipartite match associates each ground-truth trajectory to exactly one computed trajectory by minimizing the number of mismatched frames over all the available data.
- **IDF1 score**: Ratio of correctly identified detections over the average number of ground-truth and computed detections.

$$IDF_1 = \frac{2\,IDTP}{2\,IDTP + IDFP + IDFN}$$

E. Ristani, F. Solera, R. S. Zou, R. Cucchiara, and C. Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. ECCVW, pages 17–35, 2016.

Task 2.3: IDF1 for Multiple Object Tracking

Use the IDF1 score implementation from [py-motmetrics](#) to compute your results. Alternative: [amilan-motchallenge-devkit](#) (Matlab)

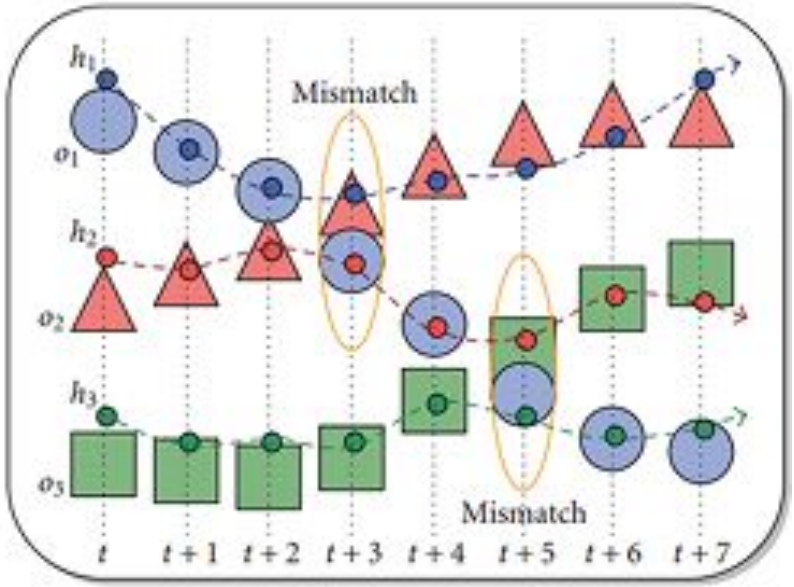


Figure from Bernardin, Keni, and Rainer Stiefelhagen. "Evaluating multiple object tracking performance: the CLEAR MOT metrics." EURASIP Journal on Image and Video Processing 2008.1 (2008): 1-10.

Task 2.3: IDF1 for Multiple Object Tracking

Team ID	T2.1 Tracking by overlap	T2.2 Tracking with a Kalman filter
Team 1		
Team 2		
Team 3		
Team 4		
Team 5		
Team 6		

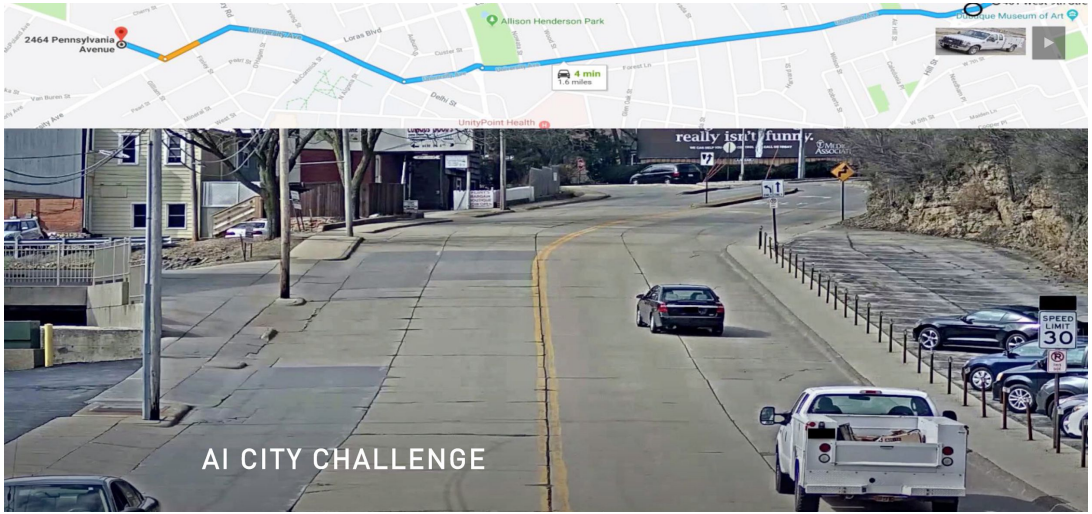
Tasks

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

(optional) Task 3: CVPR 2022 AI City Challenge

Assess the quality of your best solution on the [CVPR 2022 AI City Challenge] (Track 1):

“Participating teams will track vehicles across multiple cameras both at a single intersection and across multiple intersections spread out across a city. This helps traffic engineers understand journey times along entire corridors. The team with the highest accuracy in tracking vehicles that appear in multiple cameras will be declared the winner of this track. In the event that multiple teams perform equally well in this track, the algorithm needing the least amount of manual supervision will be chosen as the winner.”



(optional) Task 3: CVPR 2022 AI City Challenge

Read the Data & Evaluation description for Track 1.

Provided data [3.3 GB]:

- Train partition
  - Sequences S01, S03 and S04
  - Different cameras for each sequence.
- README.txt

Tasks to explore:

1. Single-camera tracking.
2. Multiple-camera tracking.

You DO NOT need to process the whole dataset to complete the task.
Start small, scale later if possible.

USAGE CONDITIONS

These data can only be used for the project in M6 in the Master in Computer Vision Barcelona 2021/2022.

You must delete these data once the module has finished.

(optional) Task 3: CVPR 2019 AI City Challenge

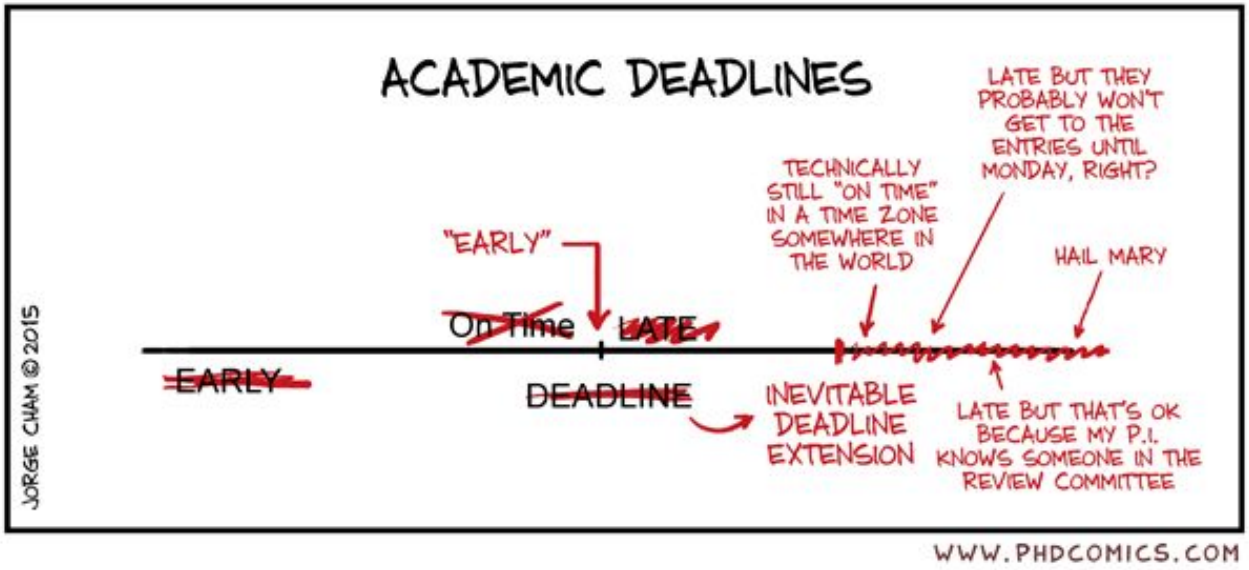
If you achieve promising results, we encourage you to participate in the CVPR 2022 AI City Challenge (Track 1):

- Challenge track submissions due: April 9.
- Workshop papers due: April 13.
- GitHub repo: April 18.

Scoring Rubric

Task	Description	Max. Score
T1	Object Detection	1
T1.1	Off-the-shelf	1
T1.2	Fine-tuning	2
T1.3	K-fold Cross validation	1
T2	Object Tracking	
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 2022 AI City Challenge	+1

Deliverables



- Deadline: **Wednesday March 30th at 3pm**.
- Deliverables:
  - Submit your report by editing these slides.
  - Provide feedback regarding the teamwork on this evaluation form.