Object detection in video using deep reinforcement learning and tracking

Anass Denguir

Outline

Outline

What is object detection?

Object detection is a computer vision problem that consists of:

- **Detecting** all the objects present in a given image. It is a regression problem which requires to find bounding boxes enclosing the objects present in the input image
- **Classifying** the localized objects by assigning them a label corresponding to their object category (person, car, cat, ...)

Object detection has a lot of applications such as:

- Smart surveillance cameras
- Self-driving cars

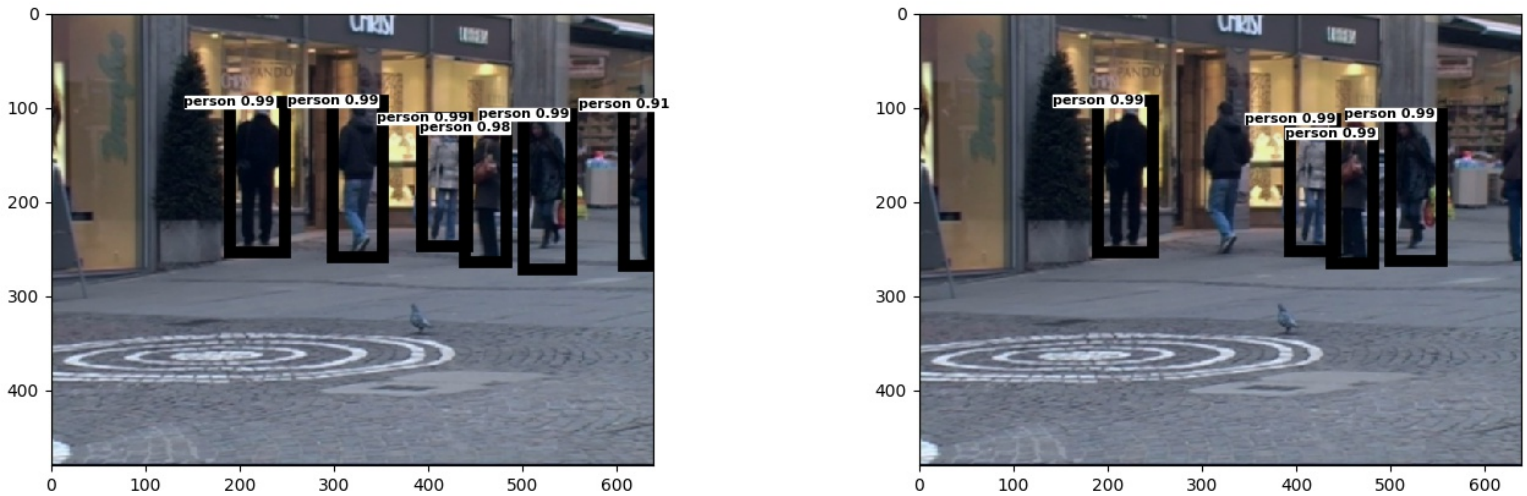
Outline

Problem

State-of-the-art object detection algorithms perform well on images.

However, their detection is not enough robust for video applications.

⟶ Previously detected objects are lost between consecutive frames

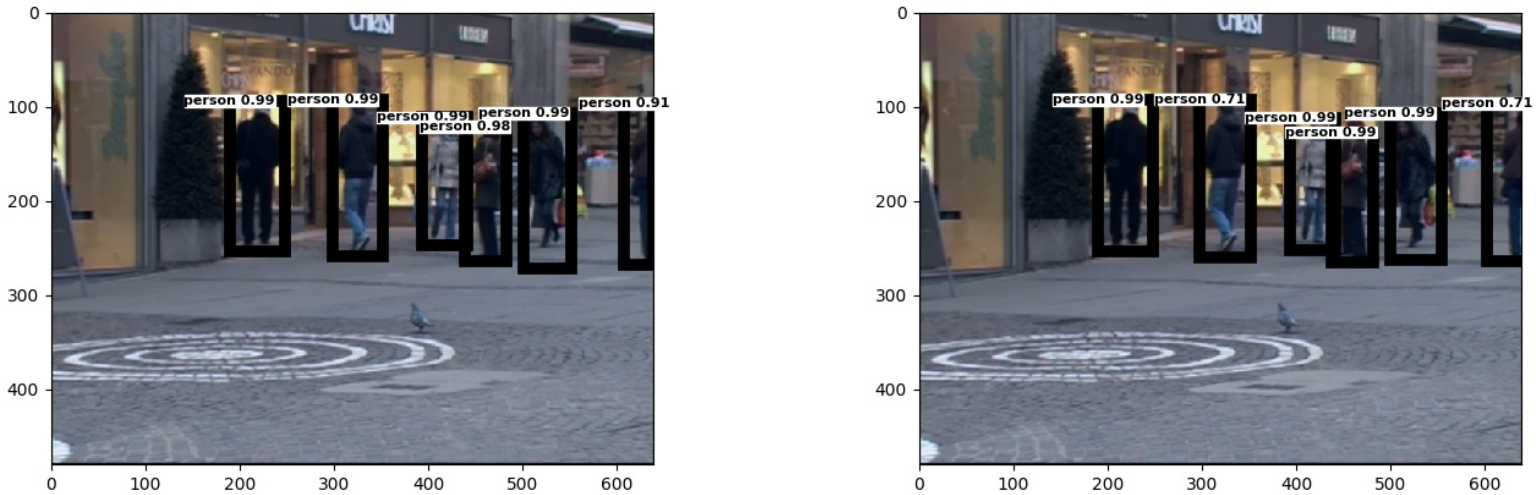


(a) frame at time $t-1$ (b) frame at time t

Figure: Problem of baseline detection method

Solution

We propose to reinforce object detection by implementing an object tracker that is responsible for predicting the motion of previously detected objects.



(a) frame at time $t - 1$

(b) frame at time t

Figure: Detection reinforced by deep-SORT tracking

Main results

The proposed method is tested on 5 videos (∼ 2500 images) extracted from MOT 2015 dataset and allows us to achieve an increase of **4%** in mean Average Precision (mAP)

	ETH-Bahnhof	ETH-Pedcross2	ETH-Sunnyday	TUD-Campus	TUD-Stadtmitte	mean
Baseline	34.69	55.34	72.5	71.05	77.85	62.29
Proposed method	**37.75**	**61.15**	**77.87**	**75**	**79.74**	**66.3**

Table: mAP results on MOT 2015 dataset

Where:

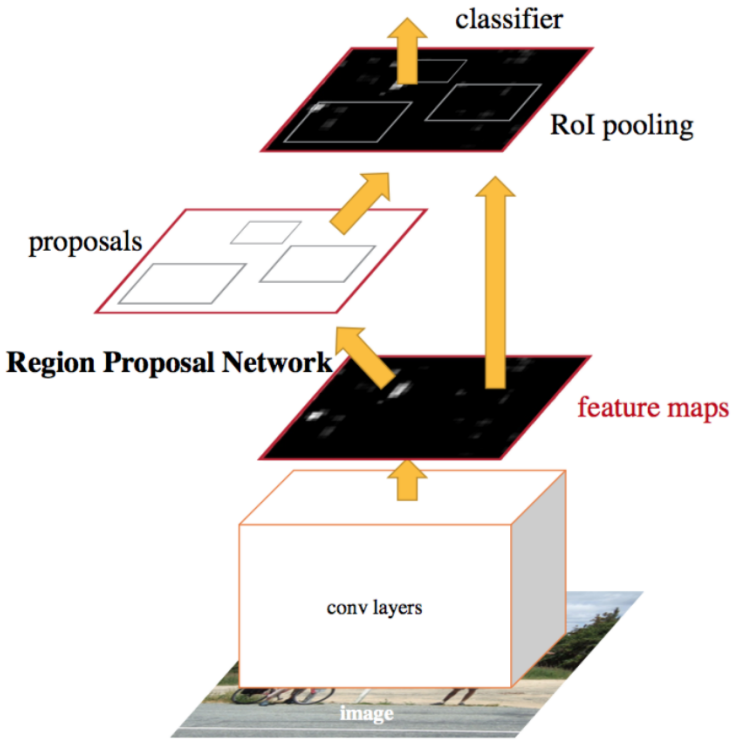
- Baseline = Object detector only
- Proposed method = Object detector + Object tracker

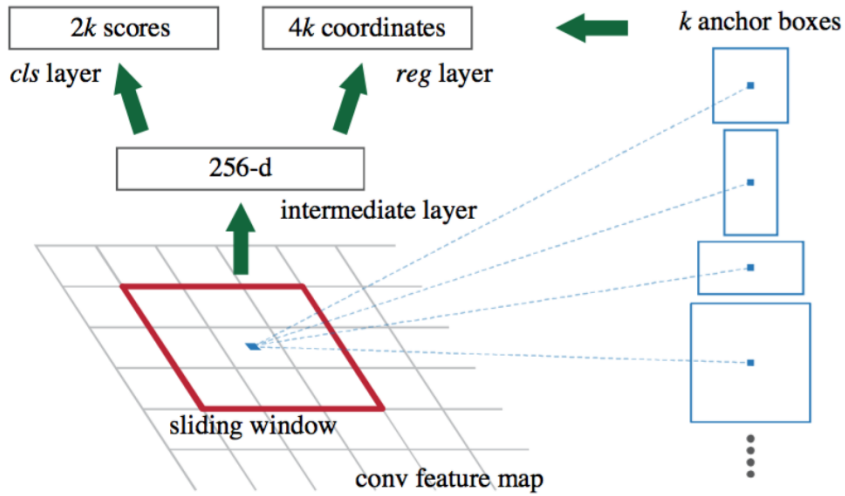
Outline

Faster R-CNN

Faster R-CNN is a deep-learning based object detector that contains 2 stages:

1. Region Proposal Network (RPN)
2. Object classifier + bounding box regressor





drl-RPN - Overview

We improve the quality of Faster R-CNN's RPN by using deep reinforcement learning. We train an agent to select the more trusted set of RoIs from the RPN ⟶ This decreases the amount of False positives in the final detection.

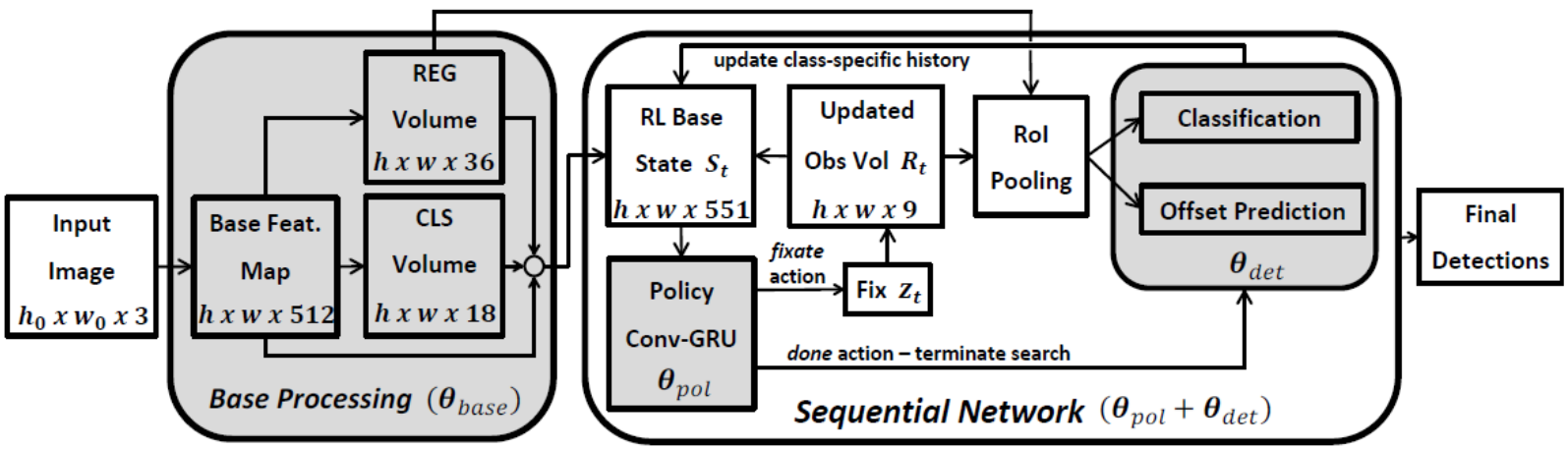


Figure: Overview of drl-RPN

drl-RPN - Visualization

Figure: Example drl-RPN

1. Get initial state S_0 from input image
2. Action: Fixate location *fix* 1
   - 2.1 Classify RoIs in fixation area
   - 2.2 Run NMS on local detections
   - 2.3 Locally detected: **person (1.00)**
   - 2.4 Update class-specific history
3. Action: Fixate location *fix* 2
   - 3.1 Classify RoIs in fixation area
   - 3.2 Run NMS on local detections
   - 3.3 Locally detected: **moto (0.98)**
   - 3.4 Update class-specific history
4. Action: Terminate search
   - 4.1 Posterior class-probability adjustments
   - 4.2 Run NMS on classified RoIs
   - 4.3 Final detections: **person** (1.00) and **moto** (0.98)

drl-RPN - States

At each time-step t:

$s_t = (S_t, H_t, R_t)$ = f(spatial context). After evaluating the state s_t, the agent issues one of the following actions:

- fixate action a_t^f: The agent fixates a new fixation window on the image. All the RoIs contained in this window are extracted.
- done action a_t^d: The agent stops the search process and classifies the RoIs collected so far

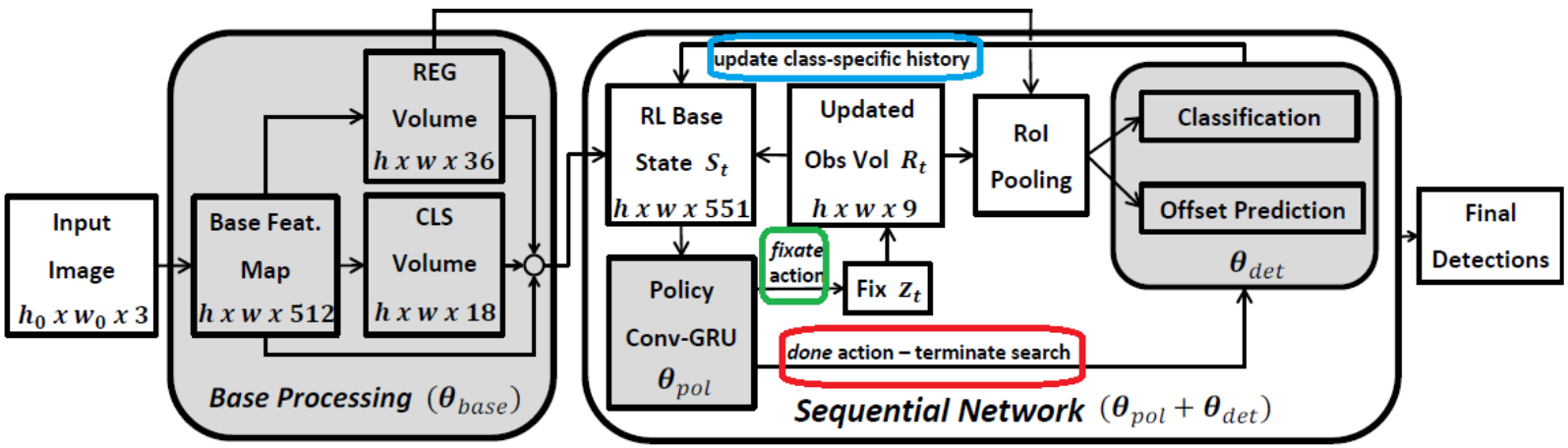


Figure: Agent states and actions

drl-RPN - Conv-GRU

- The actions of the agent are decided by following a stochastic policy π_θ, which is updated by a Conv-GRU
- At each time step t of the search process, the hidden state H_t of the agent is updated with the observation S_t and H_{t-1}
- The output of the Conv-GRU is a two-channel action volume A_t

$$
\begin{cases}
O_t &= \sigma[W_{so} * S_t + W_{ho} * H_{t-1} + b_o] \\
\widetilde{H}_t &= W_{sh} * S_t + W_{hh} * (O_t \odot H_{t-1}) + b_h \\
Z_t &= \sigma[W_{sz} * S_t + W_{hz} * H_{t-1} + b_z] \\
H_t &= (1 - Z_t) \odot H_{t-1} + Z_t \odot tanh[\widetilde{H}_t] \\
\widetilde{A}_t &= relu[W_{h\widetilde{a}} * H_t + b_{\widetilde{a}}] \\
A_t &= tanh[W_{\widetilde{a}a} * \widetilde{A}_t + b_a]
\end{cases}
\tag{1}
$$

drl-RPN - Actions

Once the action volume $A_t = \begin{bmatrix} A_t^d & A_t^f \end{bmatrix}$ determined by the Conv-GRU, the agent uses it to take a **done** or a **fixate** action.

- A **done** action is issued if $\pi_\theta(a_t^d = 1|s_t) > 0.5$, where:

$$\begin{cases} \pi_\theta(a_t^d = 1|s_t) &= \sigma[w_d^T d_t + t] \\ d_t &= vec(A_t^d) \end{cases} \tag{2}$$

- Otherwise, a **fixate** action is issued:

$$\begin{cases} \pi_\theta(a_t^d = 0, a_t^f = z_t|s_t) &= (1 - \sigma[w_d^T d_t + t])\hat{A}_t^f[z_t] \\ \hat{A}_t^f &= Softmax(A_t^f) \end{cases} \tag{3}$$

drl-RPN - Rewards

After each action, a reward r_t is attributed to the agent which evaluates the quality of the search process. The agent is trained to maximize the expected cumulative reward $J(\theta) = \mathbf{E}_{s\sim\pi_\theta}[\sum_{t=1}^{|s|} r_t]$.

- If a **fixate** action is issued:

$$r_t^f = -\beta + \sum_i \mathbb{1}[g_i : IoU_t^i > IoU^i \geq \tau] \frac{IoU_t^i - IoU^i}{IoU_{max}^i} \tag{4}$$

- If a **done** action is issued:

$$r_t^d = \sum_i \mathbb{1}[g_i : IoU_{max}^i \geq \tau] \frac{IoU^i - IoU_{max}^i}{IoU_{max}^i} \tag{5}$$

drl-RPN - Results

The parameter β allows the user to specify an exploration-accuracy trade-off according to his needs. A low value value of β will lead to

- a higher mAP
- But a longer run-time

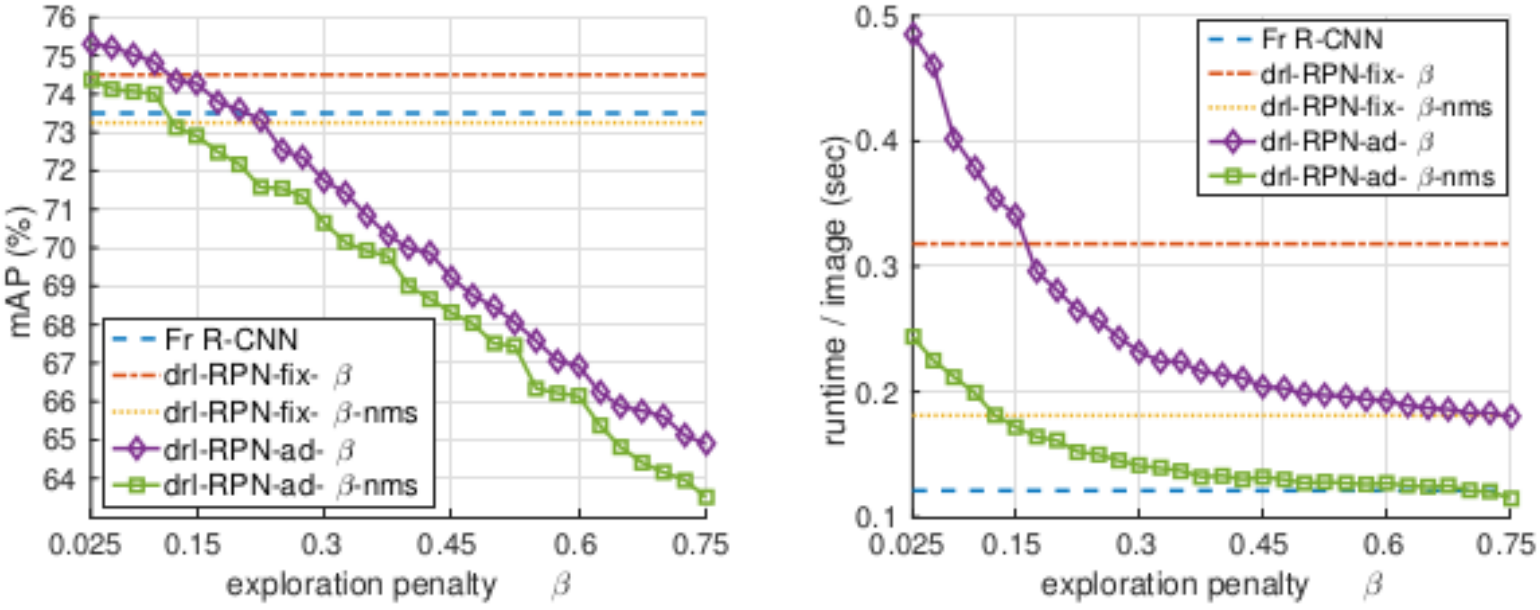


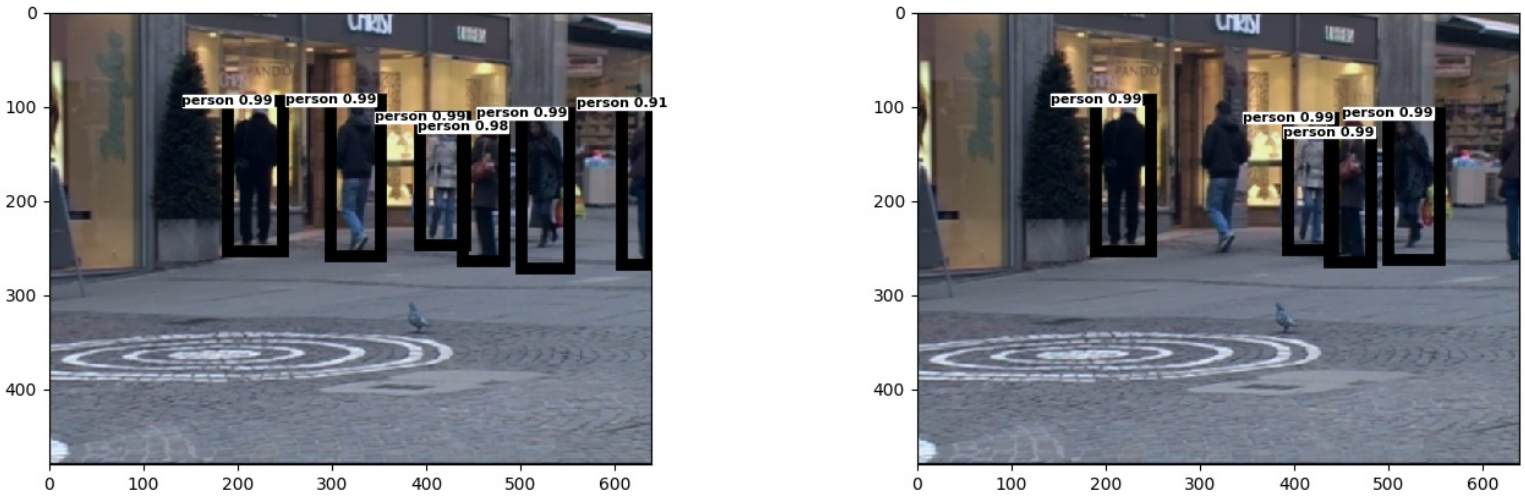
Figure: Exploration-accuracy trade-off

Outline

Problem motivation - reminder

If we use drl-RPN to detect objects on each frame without using the temporal correlation between frames, some objects are lost due to:

- Brightness changes
- Position changes



(a) frame at time $t-1$ (b) frame at time t

Figure: Problem of baseline detection method

Multiple Object Tracking

We propose to add a multiple object tracker (MOT) alongside the object detector to increase its robustness on video detection. The role of the tracker is to propagate the objects of frame $t-1$ on frame t to complete the set of object proposals.

(a) frame at time $t-1$

(b) frame at time t

Figure: Detection reinforced by deep-SORT tracking

Deep-SORT - Overview

- The performance of an object tracker highly depends on the object detector. This is why a high quality object detector like Faster R-CNN or drl-RPN is needed
- Deep-SORT is a simple and fast object tracker that tracks each detected object in an image. The tracking task is performed in 2 steps:
    1. **Motion estimation**: the future position of each object is predicted using **Kalman filtering**
    2. **Data association**: the correspondence between each tracklet and the previous detections is made using **Hungarian algorithm**

Deep-SORT - Motion estimation (1/3)

The state of each object can be described by a 7-dimensional vector x

$$x = \begin{bmatrix} u & v & s & r & \dot{u} & \dot{v} & \dot{s} \end{bmatrix}^T \tag{6}$$

Where:

- u is the horizontal coordinate of the center of the bounding box
- v is the vertical coordinate of the center of the bounding box
- s is the scale of the bounding box
- r is the aspect ratio of the bounding box

Deep-SORT - Motion estimation (2/3)

The state evolution is assumed to follow a linear model:

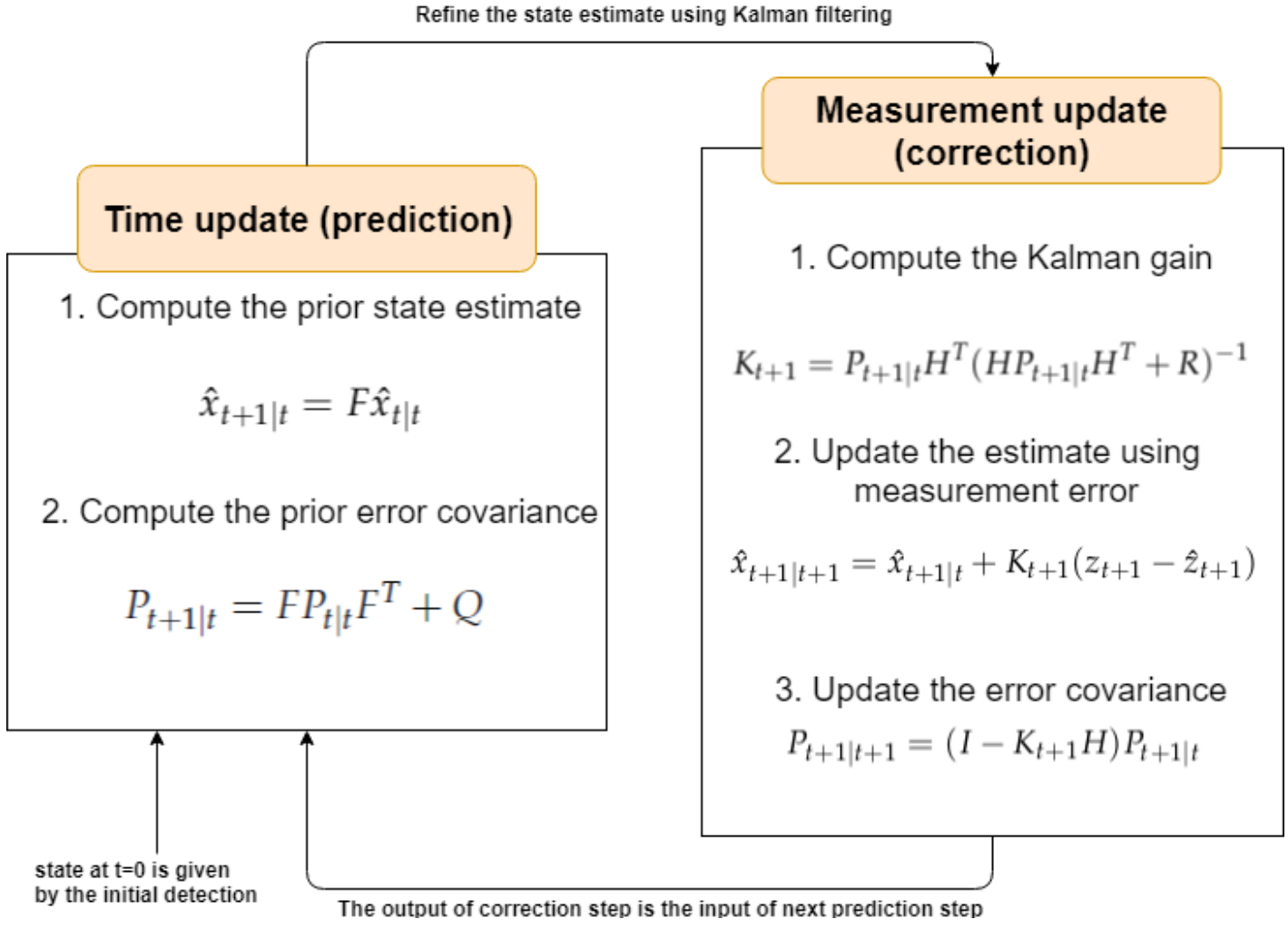
$$\begin{cases} x_{t+1} &= Fx_t + w_t \\ z_{t+1} &= Hx_{t+1} + n_{t+1} \end{cases} \tag{7}$$

Where:

- F is the state transition matrix
- H is the measurement matrix
- w_t is a white Gaussian noise with covariance $Q = E[w_t w_t^T]$
- n_t is a white Gaussian noise with covariance $R = E[n_t n_t^T]$

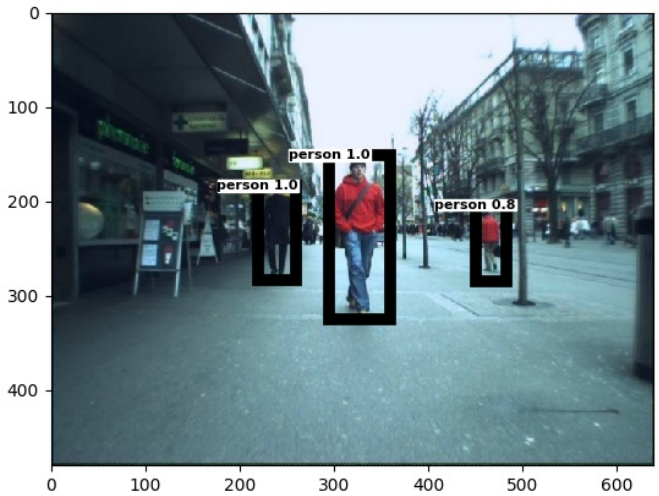
Deep-SORT - Motion estimation (3/3)

The goal is to find the state estimate $\hat{x}_{t+1}$ that minimizes the error covariance $P_{t+1|t+1} = E[(x_{t+1} - \hat{x}_{t+1|t+1})^2]$

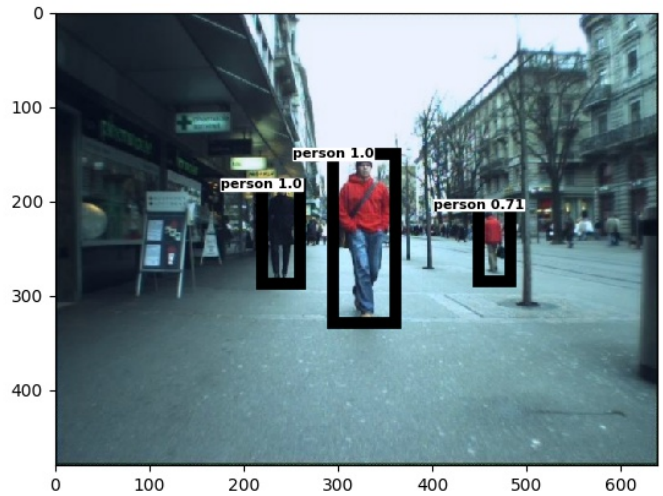


Deep-SORT - Data association (1/3)

Data association consists of associating each predicted box of frame $t + 1$ to the corresponding boxes of frame t



(a) Detection at time t



(b) Prediction for time $t + 1$

Figure: Tracking example

Deep-SORT - Data association (2/3)

The data association problem is solved by minimizing the cost matrix C, which evaluates the distance between each pair (i, j). The cost matrix is built by balancing two different distances $d^{(1)}$ and $d^{(2)}$

$$C(i,j) = \lambda d^{(1)}(i,j) + (1-\lambda) d^{(2)}(i,j) \tag{8}$$

The first distance is based on the Intersection over Union (IoU):

$$d^{(1)}(i,j) = -IoU(i,j) \tag{9}$$

$$d^{(1)} = -\begin{bmatrix} IoU(d1,t1) & IoU(d1,t2) & IoU(d1,t3) \\ IoU(d2,t1) & IoU(d2,t2) & IoU(d2,t3) \\ IoU(d3,t1) & IoU(d3,t2) & IoU(d3,t3) \end{bmatrix} \tag{10}$$

Deep-SORT - Data association (3/3)

The second distance metric $d^{(2)}$ is a deep cosine metric that measures the appearance similarity

$$d^{(2)}(i,j) = min\{1 - r_j^T r_k^{(i)} | r_k^{(i)} \in R_i\} \tag{11}$$

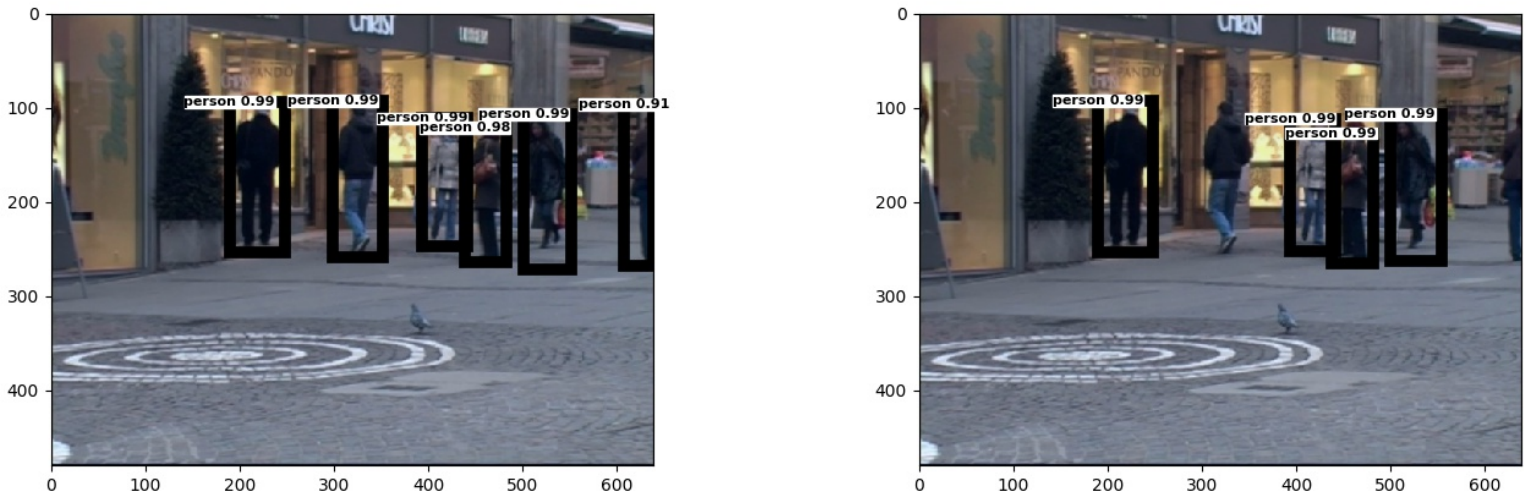
Where:

- r_j is a 128-d appearance vector that describes the detected object j in frame t
- $r_k^{(i)} \in R_i$ is a gallery of the last 100 appearance vectors corresponding to the tracklet i.

$\longrightarrow$ The deep cosine metric is useful for detecting occluded objects because it is not based on small motion assumption

The proposed method (1/2)

We need an object tracker that performs well at tracking occluded objects because the objects that are missed by the object detector can be considered as occluded in the eyes of the detector.



(a) frame at time $t-1$ (b) frame at time t

Figure: Problem of baseline detection method

The proposed method (2/2)

For each frame t, we propose to combine the tracklets of deep-SORT with the detection of drl-RPN to increase detection accuracy (mAP):

Algorithm 1 Combined detection and tracking

input: frame sequence $\{I_t\}_{t=0}^{t_{end}}$

$D_0 := DetectOnImage(I_0)$

initialize the tracklets T_0 from D_0

1: **for** t = 1 **to** t_{end} **do**
2: $D_t := DetectOnImage(I_t)$
3: $T_t := PropagateBox(T_{t-1}, D_t)$
4: $T_t := RescoreBox(T_t)$
5: $D_t := D_t \cup T_t$
6: $D_t := NMS(D_t)$
7: **end for**

output: All detected boxes $\{D_t\}_{t=0}^{t_{end}}$

Outline

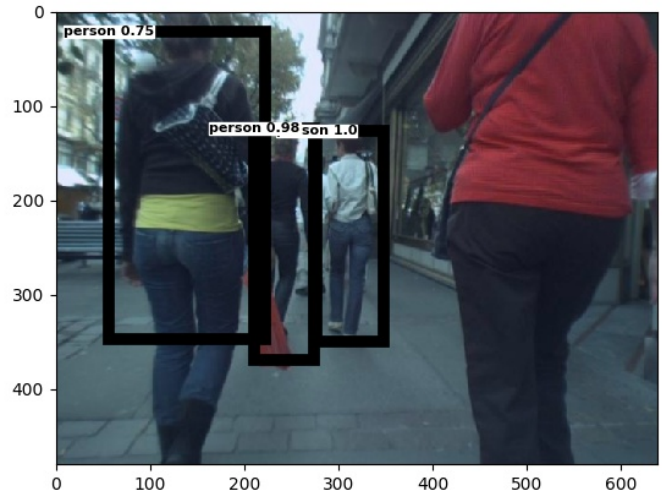
Algorithm 1 - mAP results

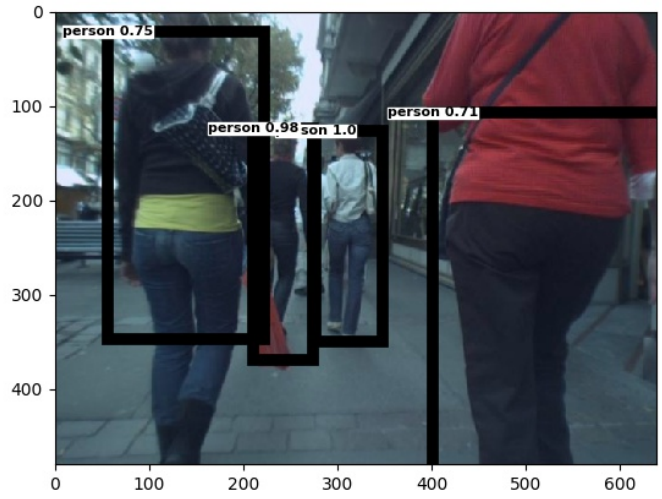
The proposed method is tested on 5 videos extracted from MOT 2015 dataset. One can see that the proposed method (last line) increases the mAP of the baseline method by more than **4%**

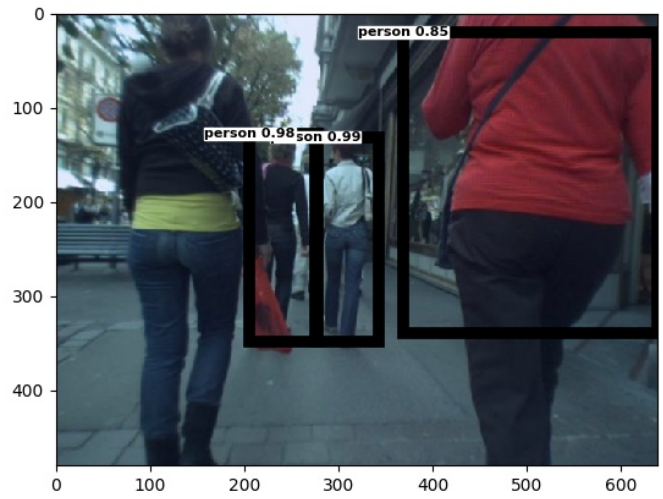
	ETH-Bahnhof	ETH-Pedcross2	ETH-Sunnyday	TUD-Campus	TUD-Stadtmitte	mean
drl-RPN (base-line)	34.69	55.34	72.5	71.05	77.85	62.29
drl-RPN +optical flow	34.29	56.21	72.1	69.86	77.99	62.09
drl-RPN +SORT	34.68	55.34	72.5	71.05	77.85	62.28
drl-RPN +deep-SORT	**37.75**	**61.15**	**77.87**	**75**	**79.74**	**66.3**

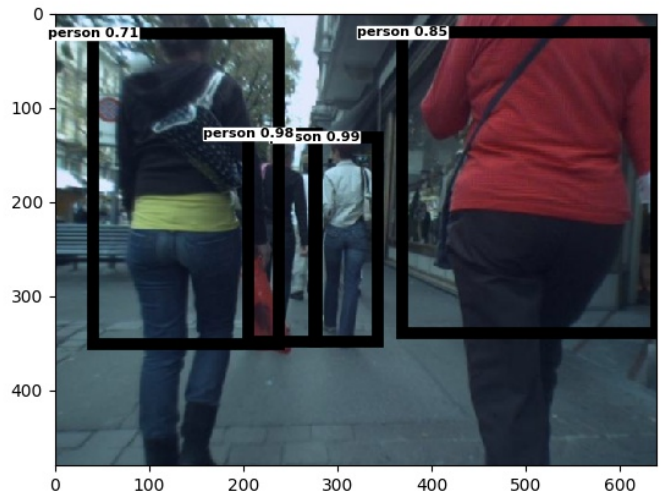
Table: mAP results using Algorithm 1 on MOT 2015 dataset

Detection results - Baseline vs Ours (1/6)

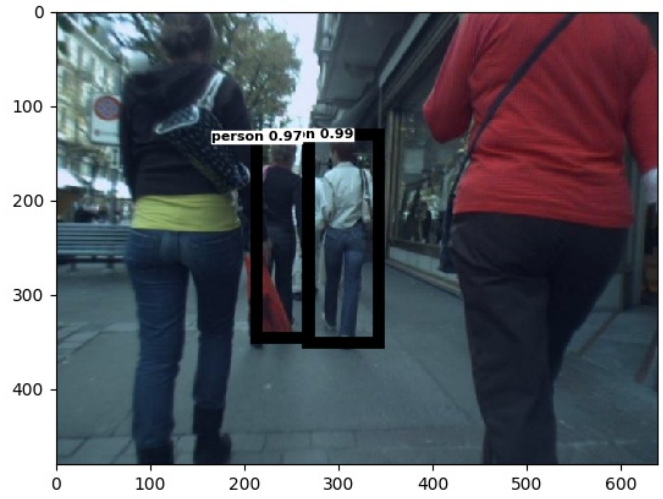


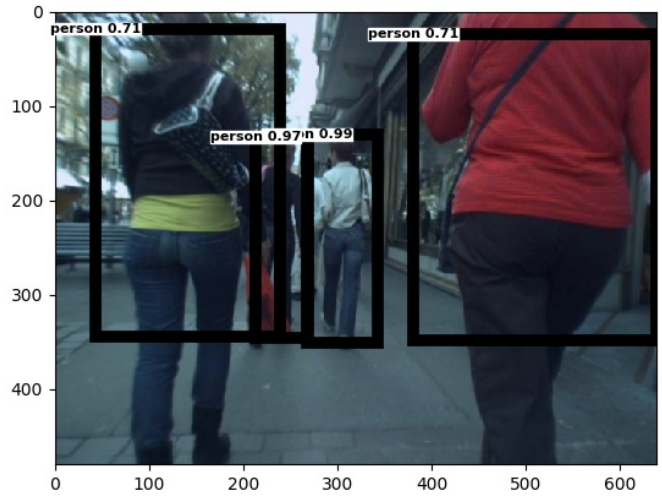


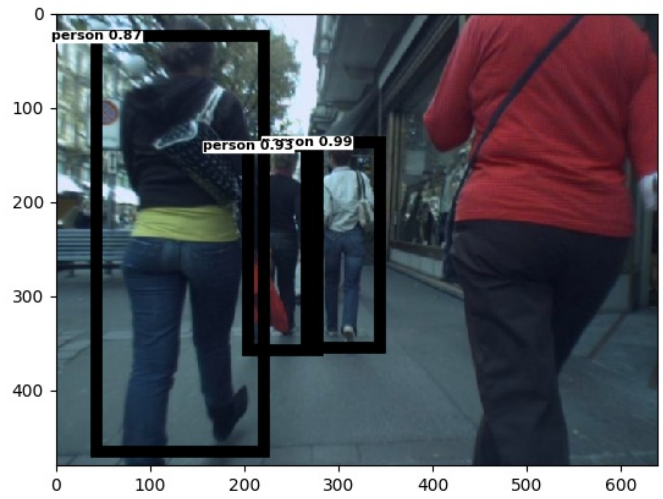


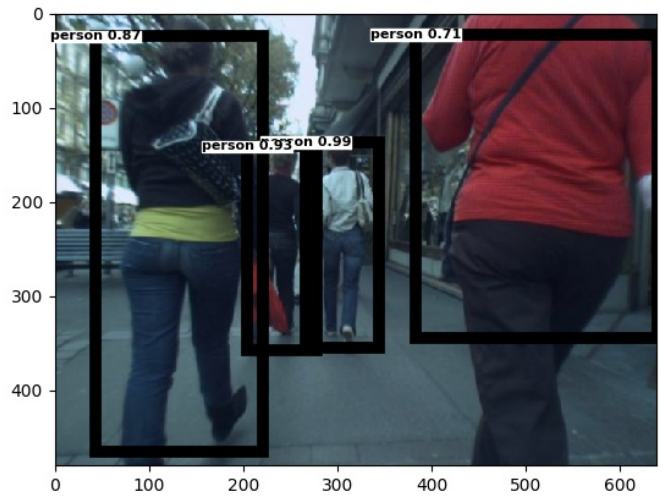


Detection results - Baseline vs Ours (2/6)

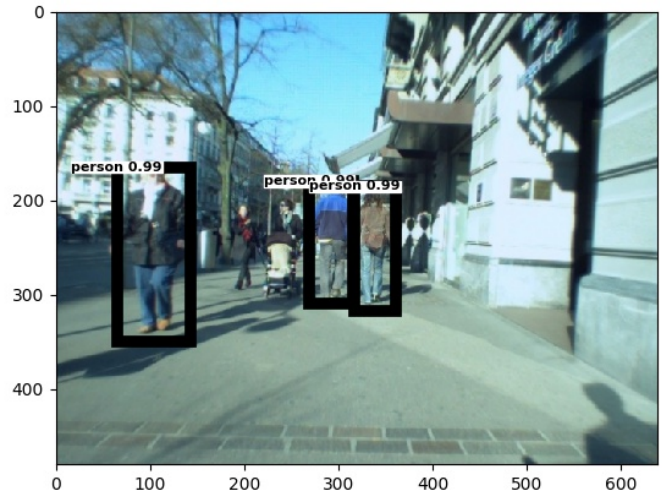


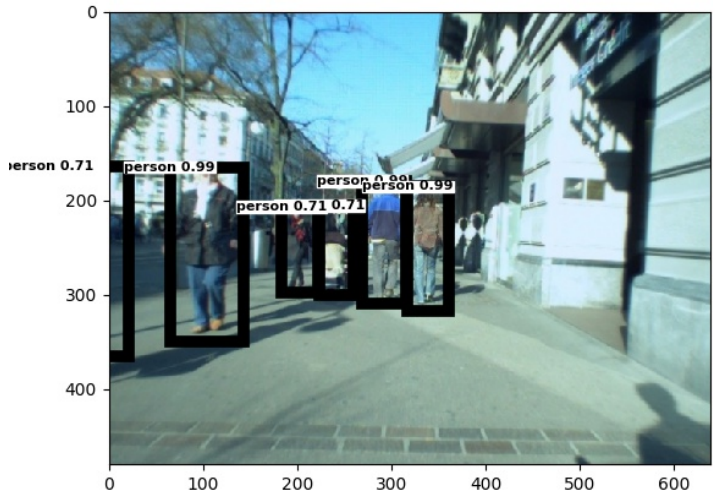


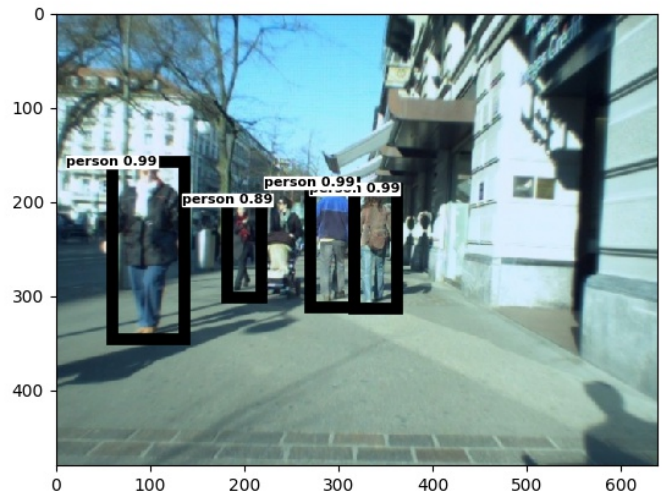


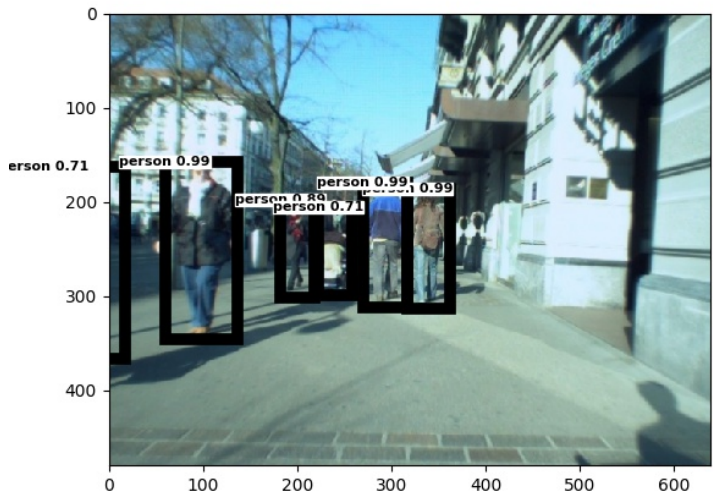


Detection results - Baseline vs Ours (3/6)

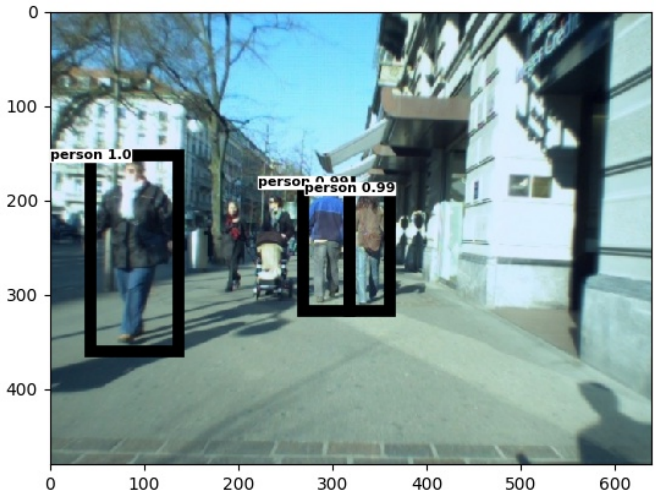


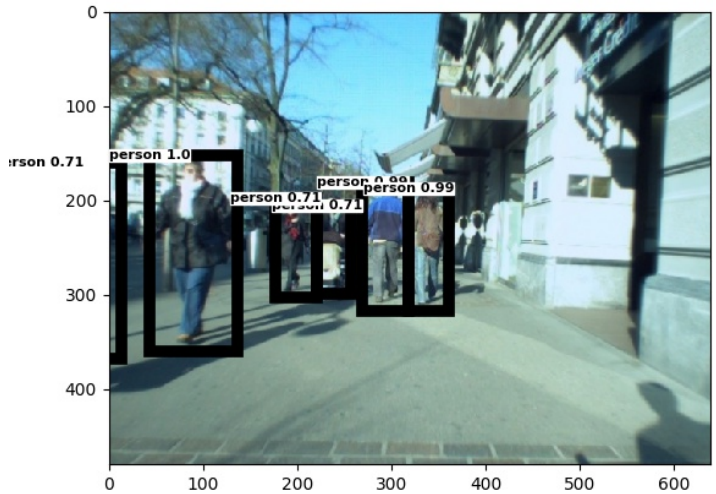


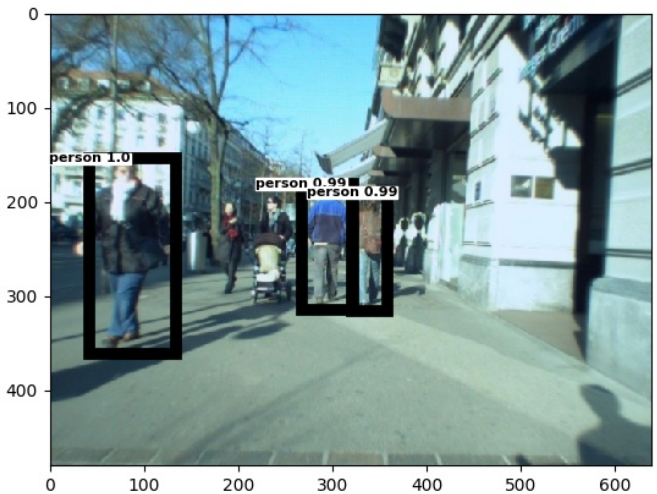


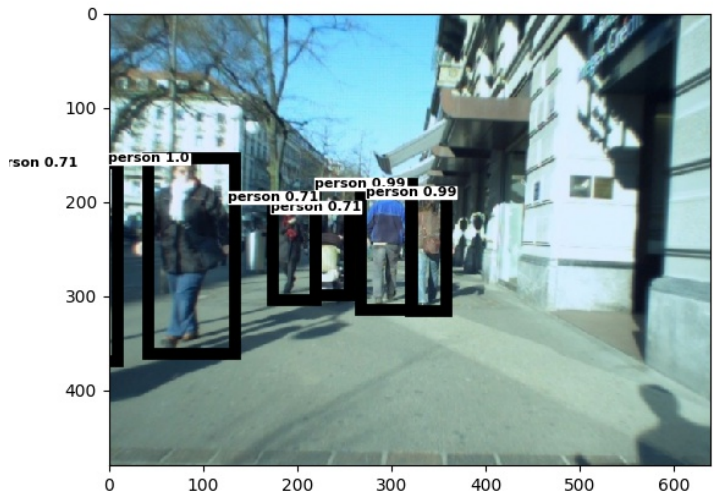


Detection results - Baseline vs Ours (4/6)

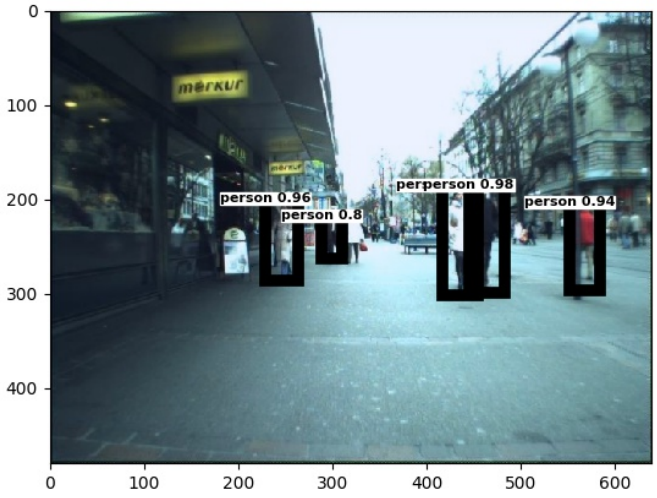


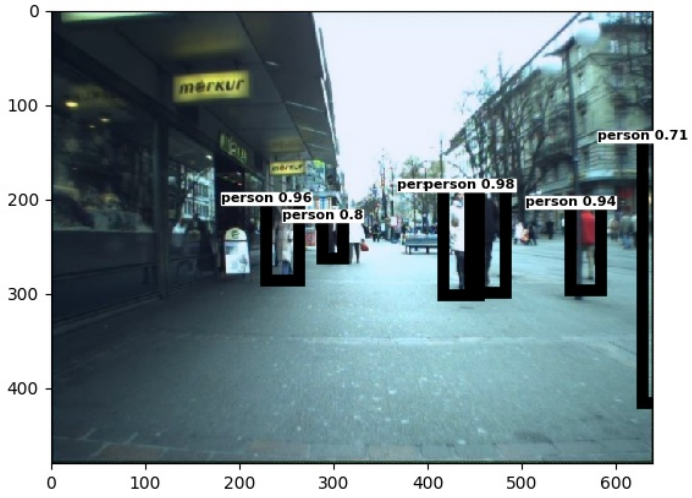


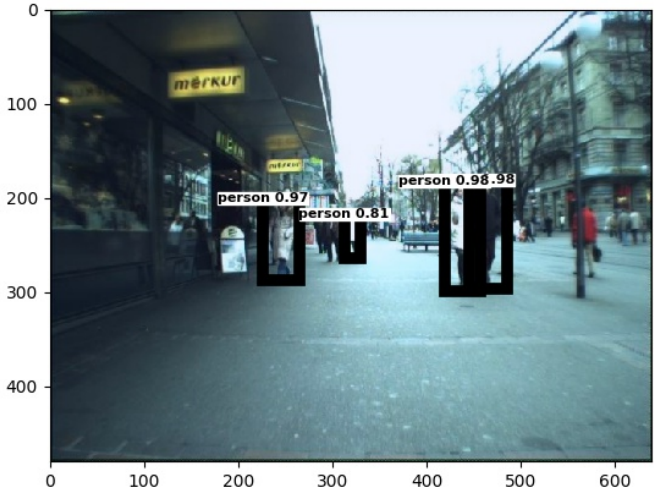


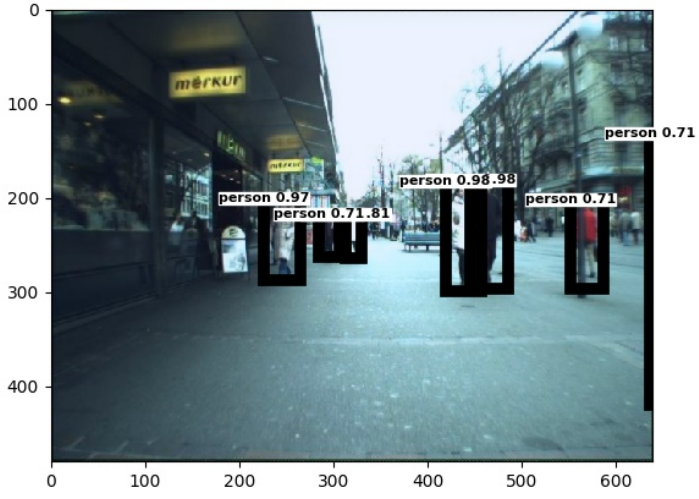


Detection results - Baseline vs Ours (5/6)

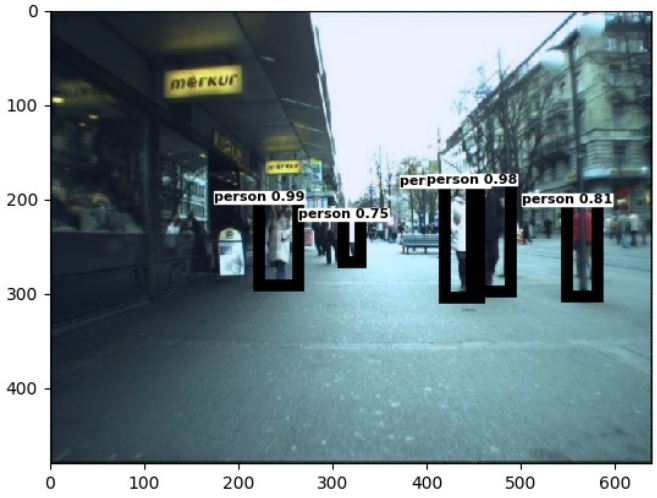


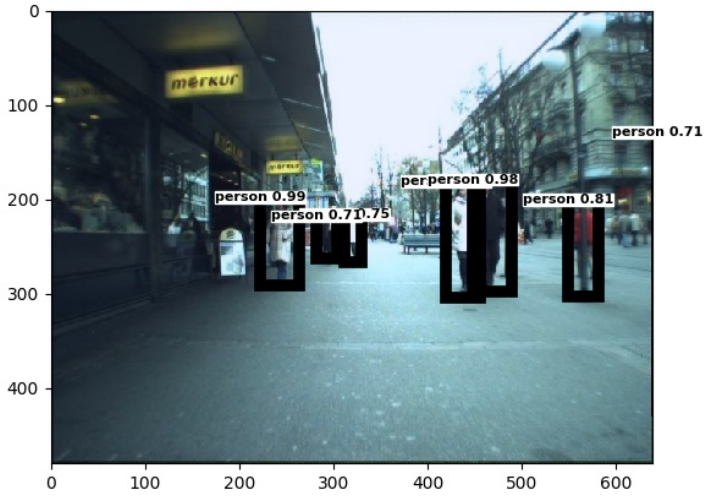


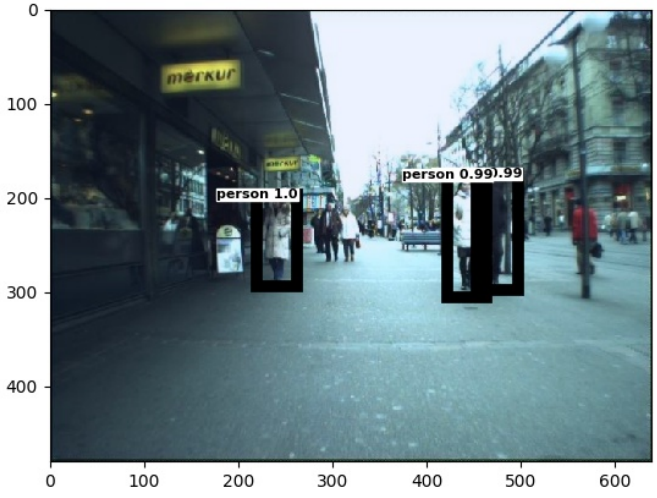


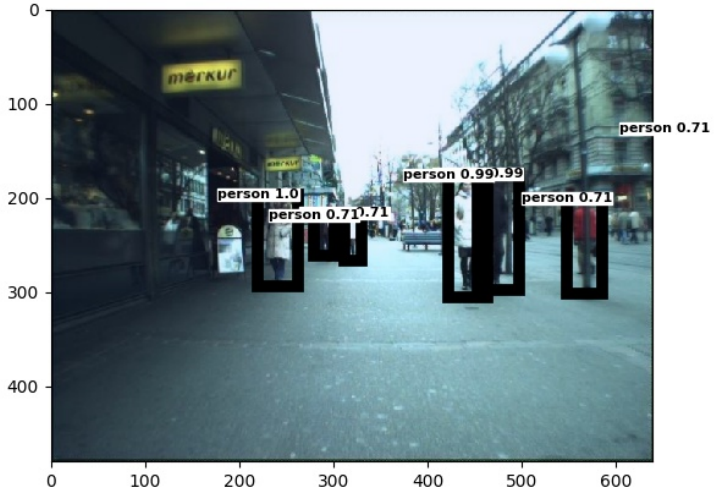


Detection results - Baseline vs Ours (6/6)









Outline

Conclusion

- Drl-RPN is a new Region Proposal Network that uses deep reinforcement learning to improve the detection accuracy of any two-stage object detector
- However, simple object detector are not enough robust for video detection because the bounding boxes are unstable
- We introduced an object tracker to take into account the temporal correlation of video. The object tracker completes the object proposals of the object detector to increase the detection accuracy

Outline

Future work

- In this work, we proposed a late integration of the tracklets to help the object detection
- As a future work, we propose to use earlier the tracklets to improve the RPN and also to use them to influence the scoring and the bounding box regression of the object proposals

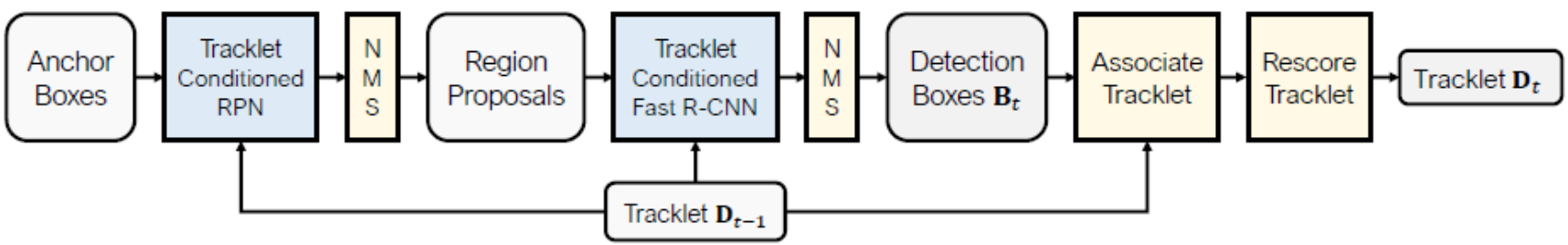


Figure: Tracklet-conditioned two-stage detector

Thank you for your attention

Appendix

Exploiting the SORT run-time

If the user is interested in increasing processing speed rather than detection accuracy, it is possible to boost the video processing by a speed-up factor **n**:

Algorithm 2 Alternated detection and tracking

input: frame sequence $\{I_t\}_{t=0}^{t_{end}}$

for t = 0 **to** t_{end} **do**
  if $t < m$ **or** $t \mod n = 0$ **then**
    $D_t := DetectOnImage(I_t)$
    $T_t := PropagateBox(T_{t-1}, D_t)$
  else
    $T_t := PropagateBox(T_{t-1})$
    $T_t := RescoreBox(T_t)$
    $D_t := T_t$
  end if
end for

output: All detected boxes $\{D_t\}_{t=0}^{t_{end}}$

Algorithm 2 - Runtime results

The table below reports the run-time of the second algorithm that aims at improving the processing speed of the detection process:

	ETH-Bahnhof	ETH-Pedcross2	ETH-Sunnyday	TUD-Campus	TUD-Stadtmitte	mean
drl-RPN (base-line)	0.908	1.048	0.871	1.105	1.116	1.01
drl-RPN +SORT (n=2)	0.461	0.543	0.449	0.617	0.590	0.532
drl-RPN +SORT (n=3)	0.317	0.370	0.307	0.462	0.430	0.377
drl-RPN +SORT (n=4)	0.241	0.287	0.236	0.383	0.333	0.296
drl-RPN +SORT (n=5)	0.197	0.237	0.196	0.332	0.285	0.249

Table: Run-time (seconds) results using Algorithm 2 on MOT 2015 dataset

Algorithm 2 - Speed-up factor

The higher the speed-up factor *n*:

- the lower the run-time
- but the lower the mAP