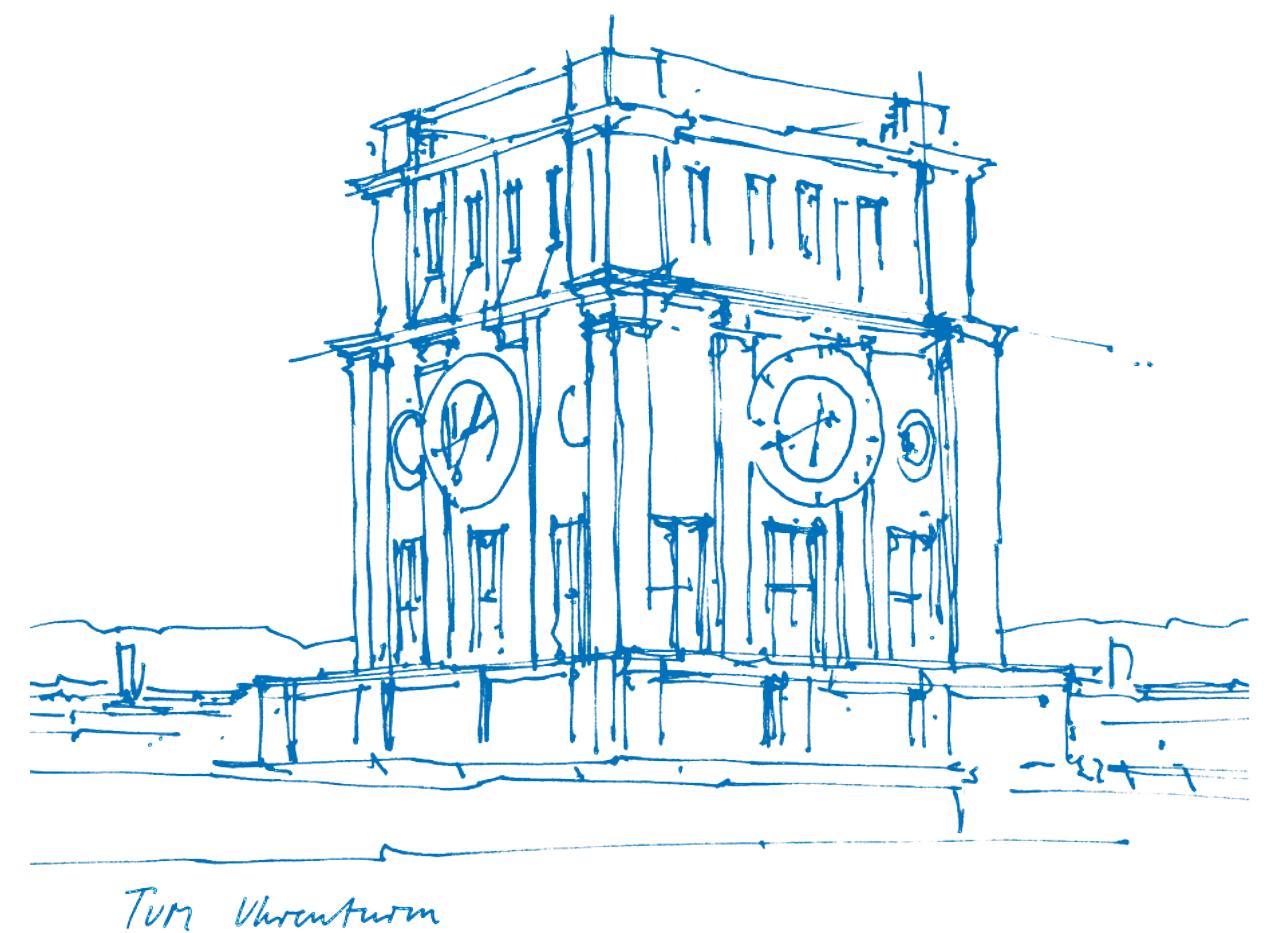


Computer Vision III:

Object tracking

Dr. Nikita Araslanov
7.11.2023

Content credit:
Prof. Laura Leal-Taixé
<https://dvl.in.tum.de>

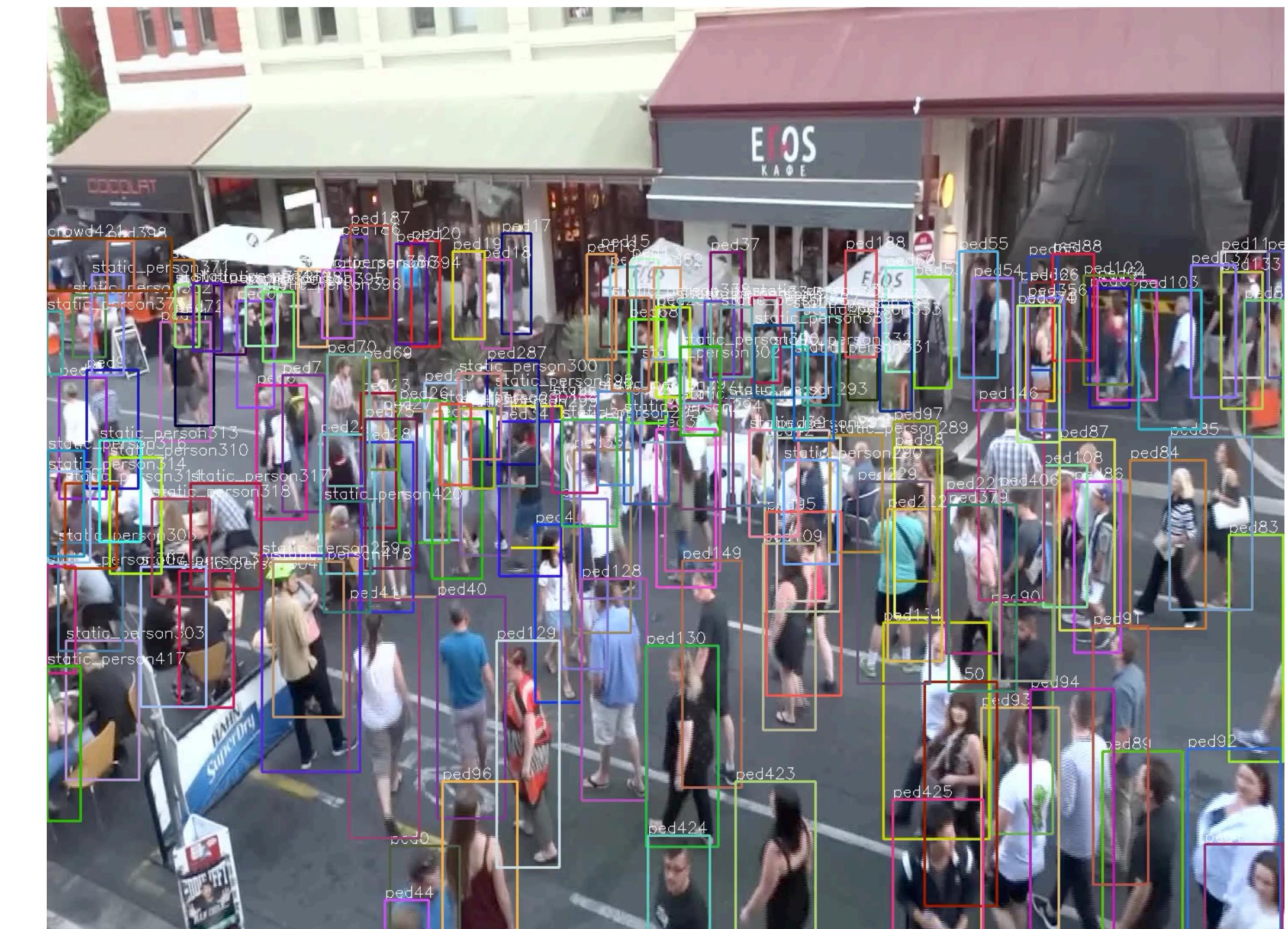


Announcements

- No lecture on November 14 (next week)
- Next lecture: November 21

Why do we need tracking?

- to build a dynamic model of the world:
 - understand where/what the objects are AND how they move;
- to forecast future motion (e.g. will the pedestrian cross the road?);
- to facilitate object detection when appearance alone is insufficient.



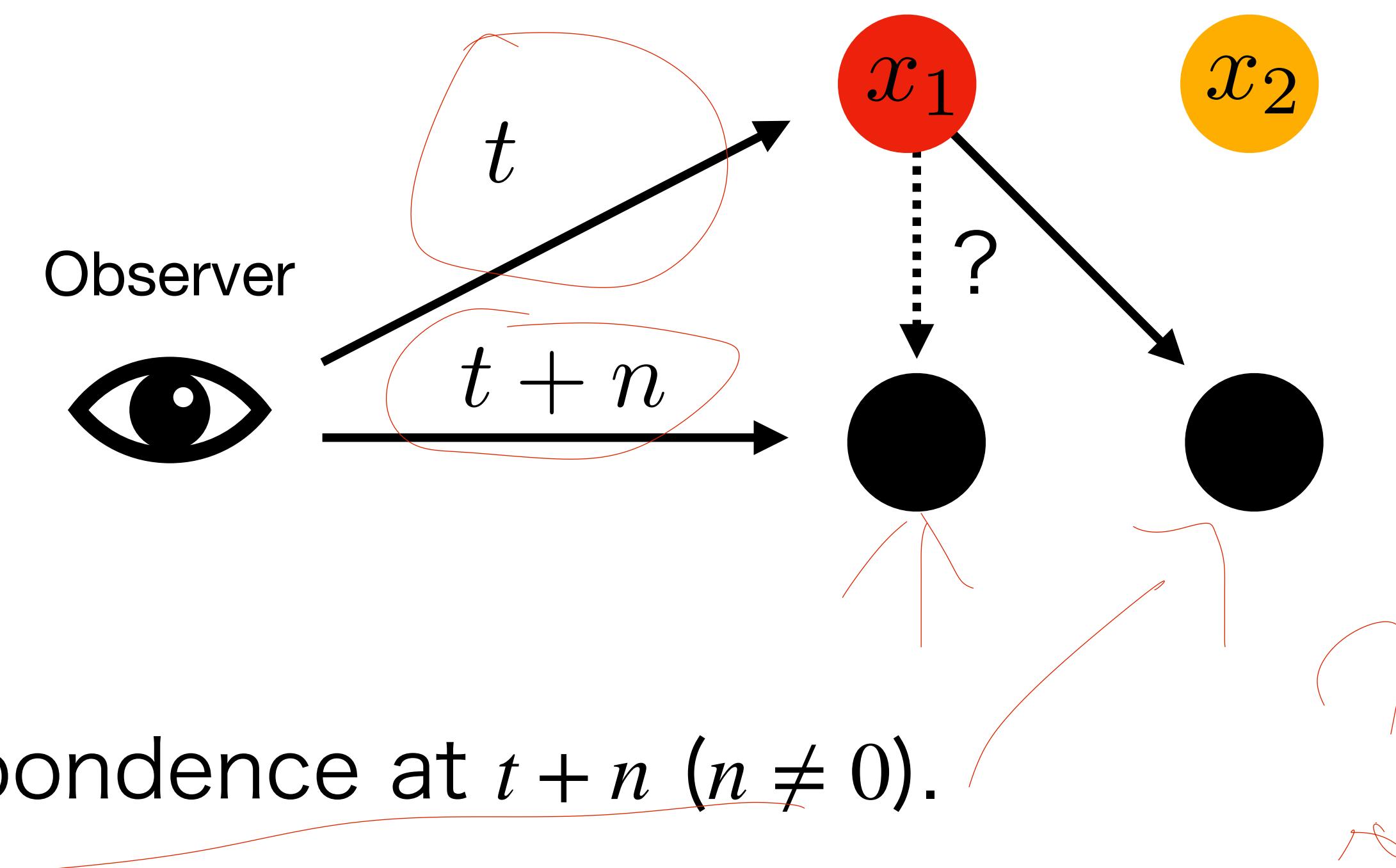
Problem statement

- Our focus: object detection and temporal association



Problem statement

Given observations at t ...



... find their correspondence at $t + n$ ($n \neq 0$).

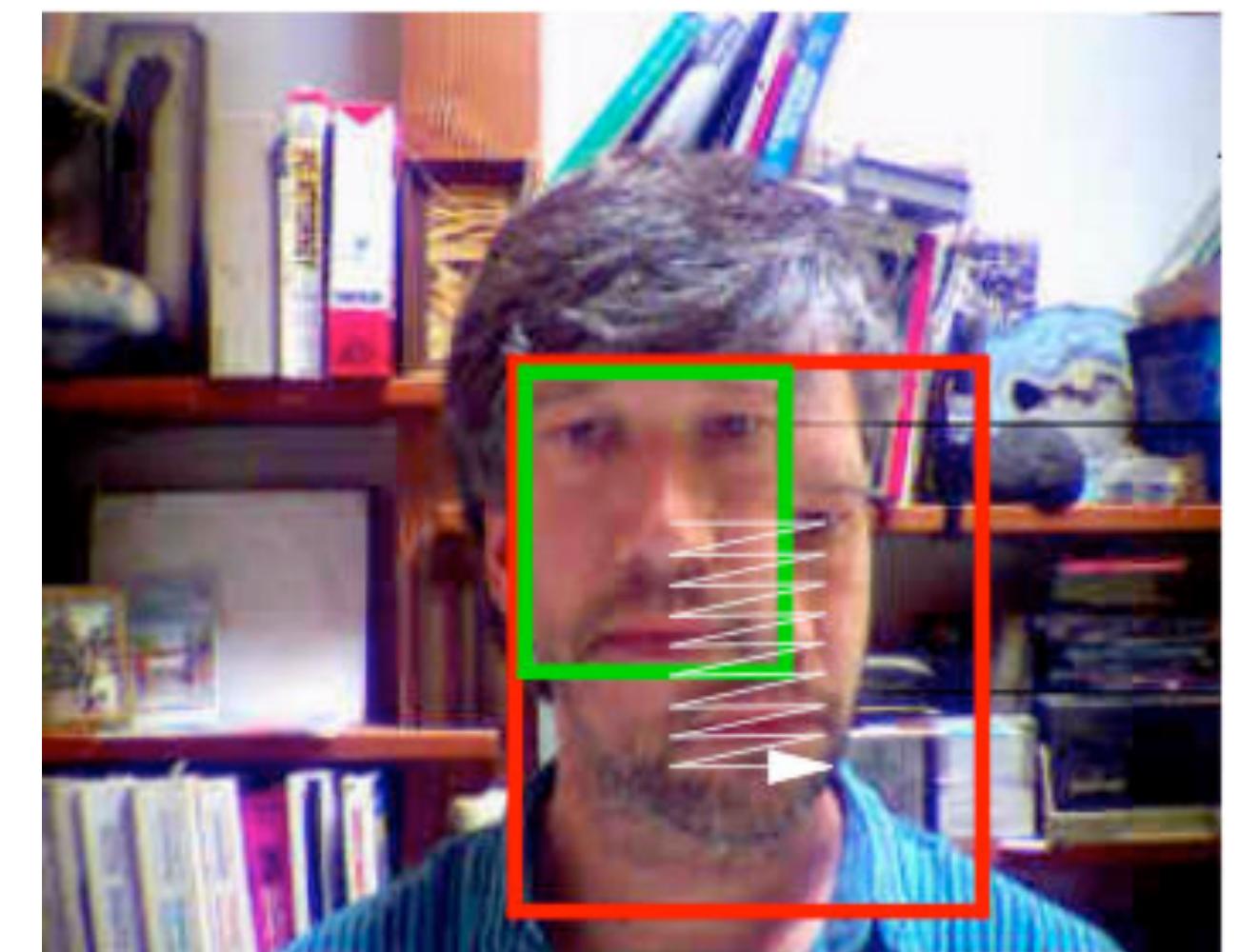
Challenges

- Fast motions
- Changing appearance
- Changing object pose
- Dynamic backgrounds
- Occlusions
- Poor image quality
- ...

[Stefan Roth]

A simple solution?

- **Tracking by detection:**
 - detect the object in every frame.
- Example: face tracking
 - If there is only one person in the scene and is always in a frontal view, we could use template matching with a sliding window (e.g. a PCA model of the face).
 - But we may not want to search the entire image, because that is inefficient.
 - Use the position from the previous frame to constrain search.

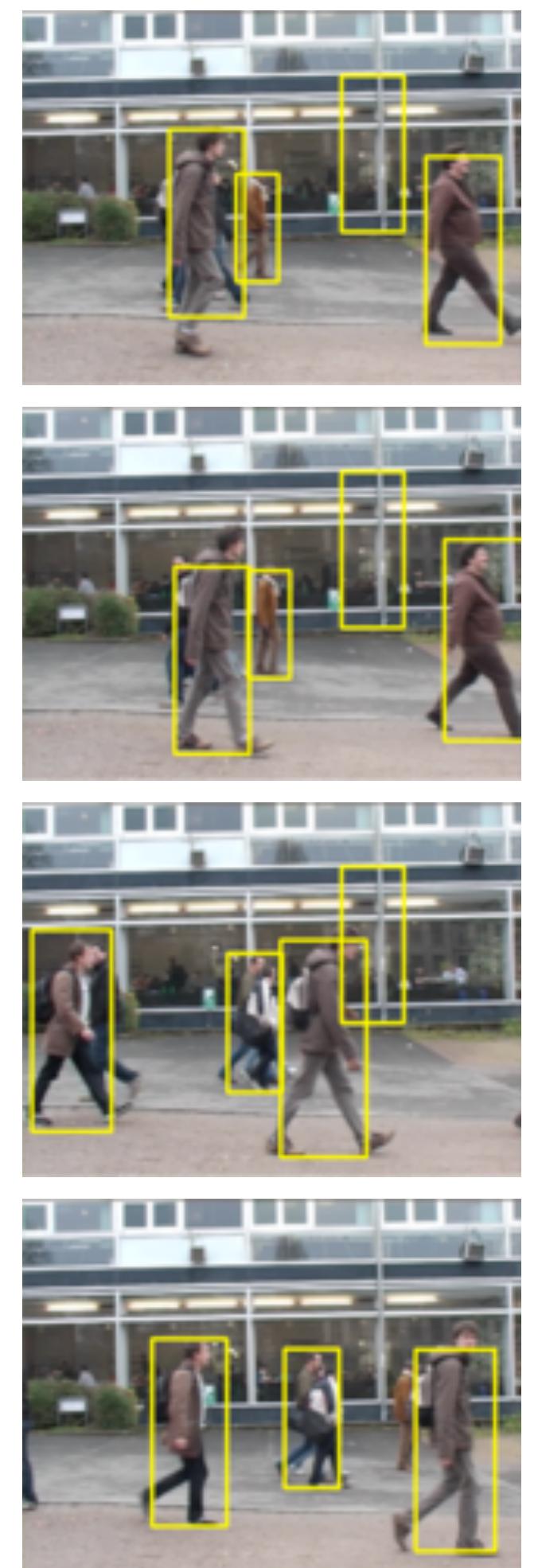


[IBM Vision Group]

[Stefan Roth]

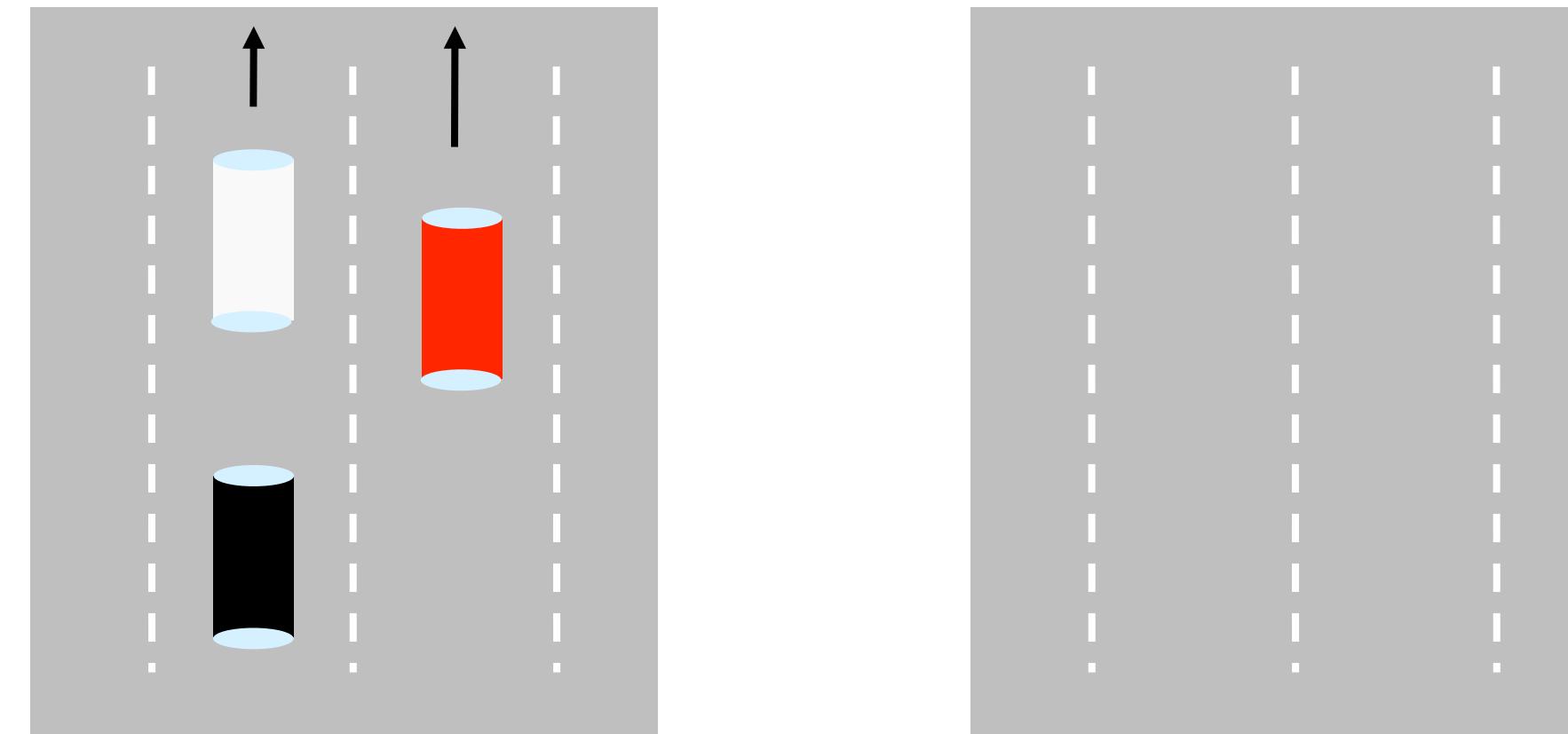
A simple solution?

- **Tracking by detection:**
 - detect the object in every frame.
- Problem: data association
 - Many objects may be present & the detector may misfire
 - How do we know which detection is which?
- We could use the motion prior.
- Let us first define a general (Bayesian) framework.



[Stefan Roth]

Illustration

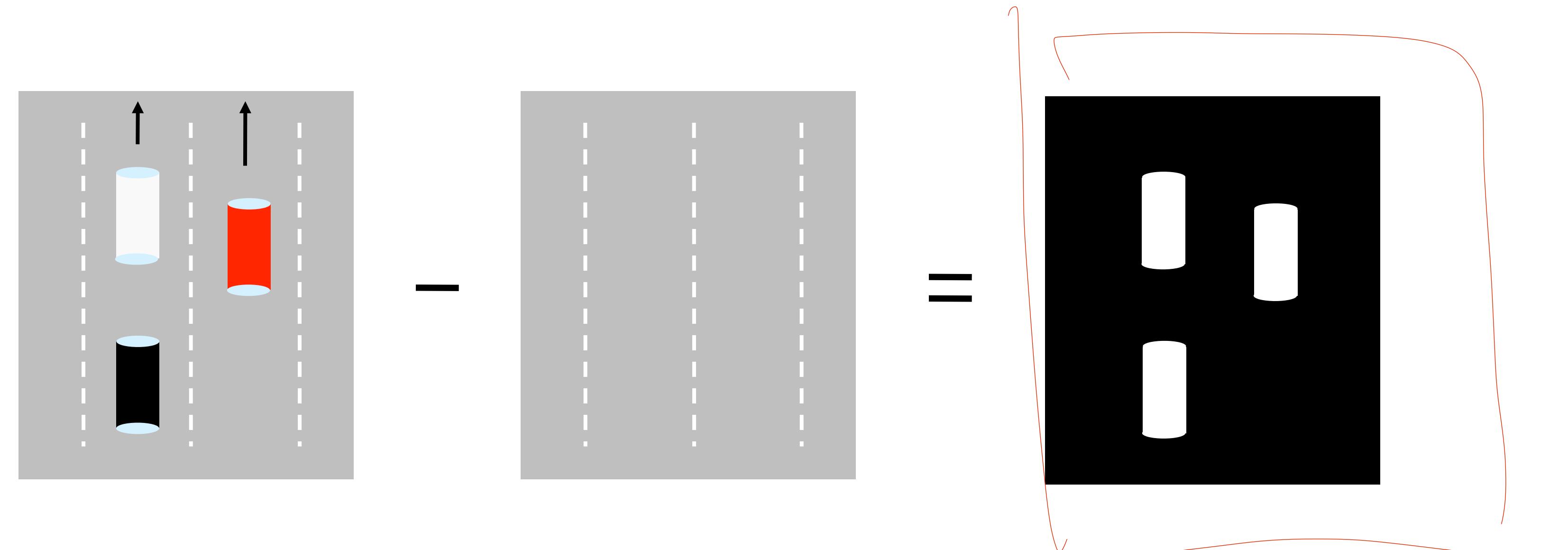


Goal: Estimate car position at each time instant (say, of the white car).

Observations: Image sequence and known background.

[Michael Black]

Illustration



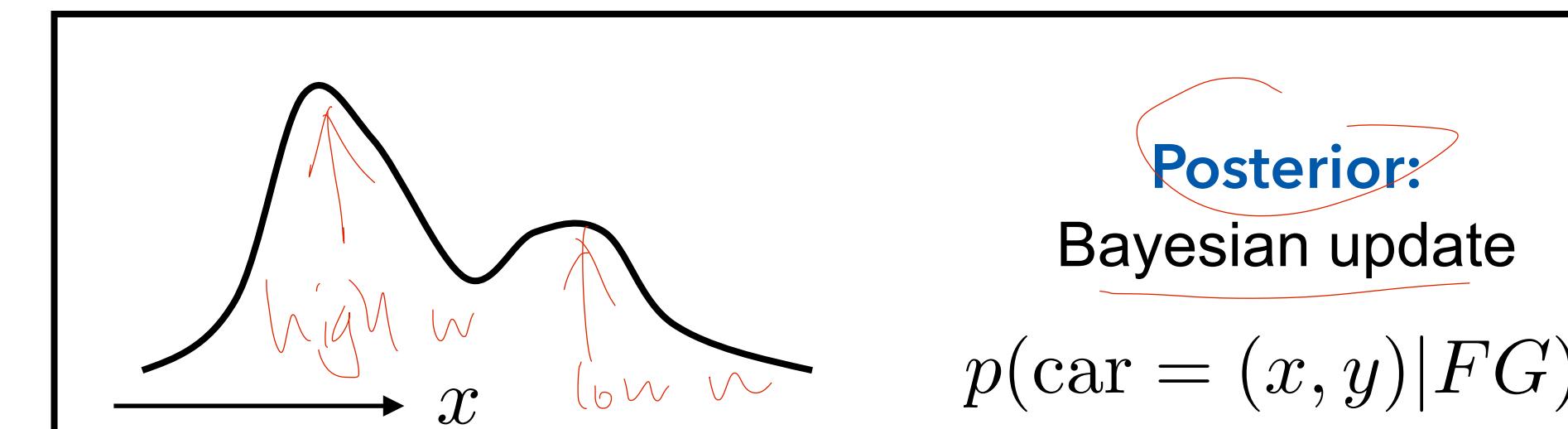
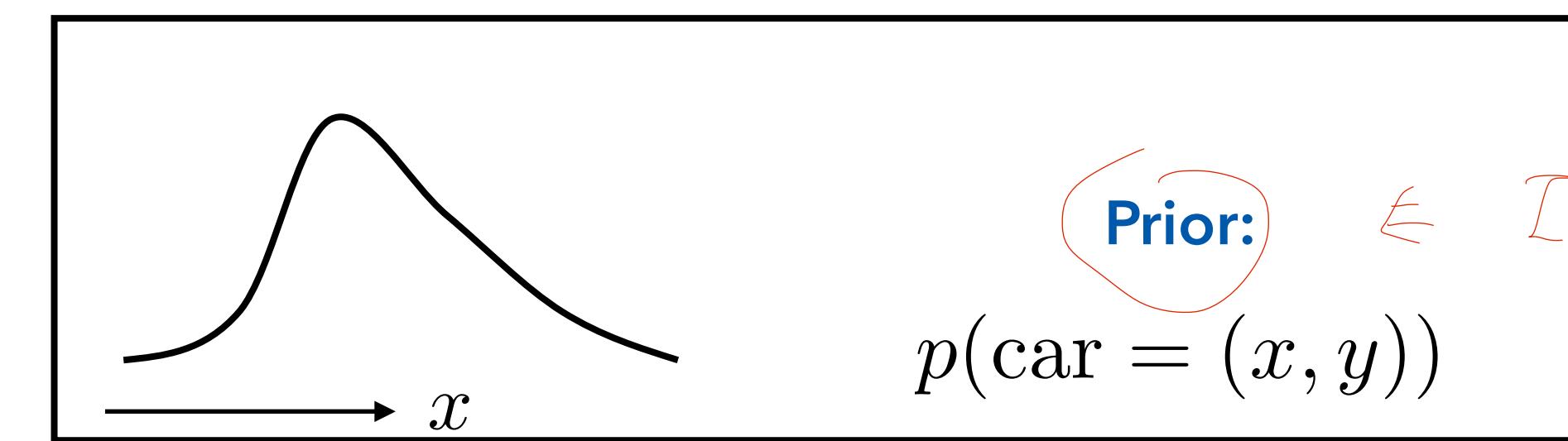
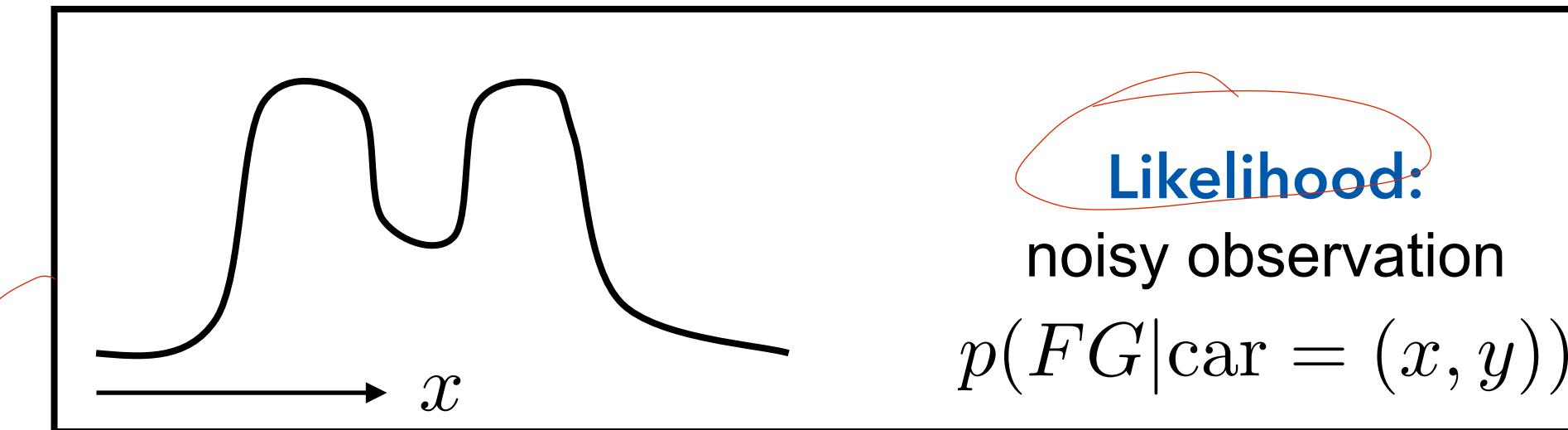
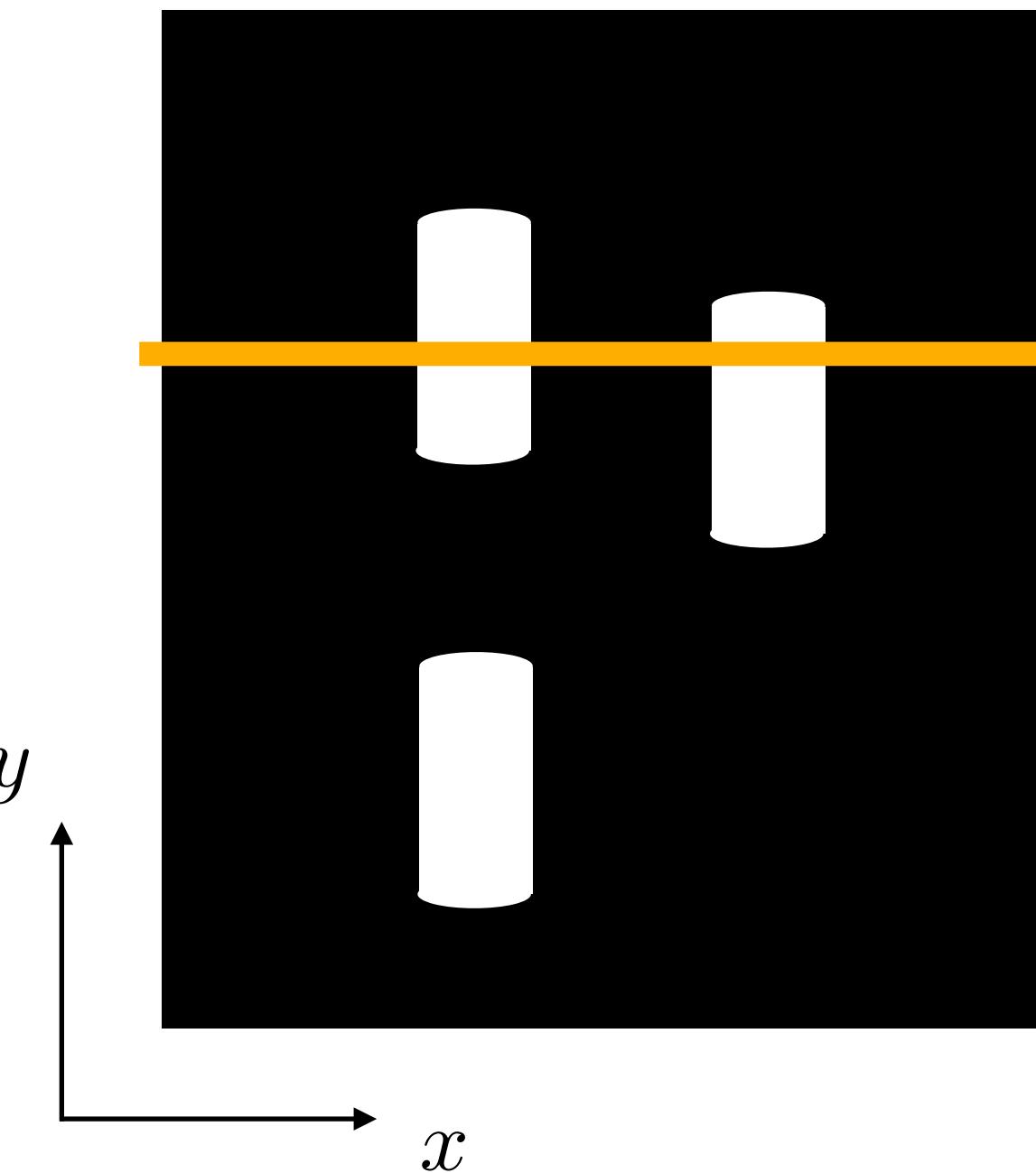
- Perform background subtraction.
- Obtain binary map of possible cars.
- But which one is the one we want to track?

[Michael Black]

Bayesian tracking

observations: images

system state: car position (x, y)



[Michael Black]

Notation

- $x_k \in \mathbb{R}^n$: **internal state** at k -th frame (hidden random variable, e.g., position of the object in the image).

$X_k = [x_1, x_2, \dots, x_k]^T$ history up to time step k .

- $z_k \in \mathbb{R}^m$: **measurement** at k -th frame (observable random variable, e.g. the given image).

$Z_k = [z_1, z_2, \dots, z_k]^T$ history up to time step k .

Goal

Estimating posterior probability $p(x_k | Z_k)$

How?

One idea: recursion

$$p(x_{k-1} | Z_{k-1}) \rightarrow p(x_k | Z_k)$$

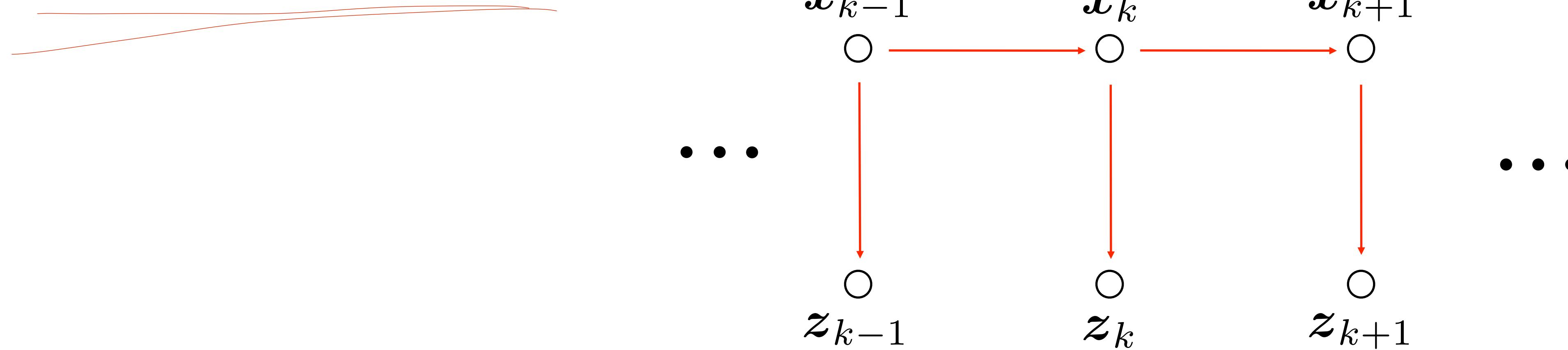
How to realise recursion?

What assumptions are necessary?

[Michael Black]

Bayesian graphical model

- Hidden Markov model:



Assumptions:

$$p(z_k | x_k, \mathbf{Z}_{k-1}) = p(z_k | x_k) \quad p(x_k | x_{k-1}, \mathbf{Z}_{k-1}) = p(x_k | x_{k-1})$$

$$p(x_k | \mathbf{X}_{k-1}) = p(x_k | x_{k-1})$$

Quiz: Name?

[Stefan Roth]

Recursive estimation

$$p(\mathbf{x}_k | \mathbf{Z}_k)$$

$$= p(\mathbf{x}_k | \mathbf{z}_k, \mathbf{Z}_{k-1})$$

$$\propto p(\mathbf{z}_k | \mathbf{x}_k, \mathbf{Z}_{k-1}) \cdot p(\mathbf{x}_k | \mathbf{Z}_{k-1})$$

$$\propto p(\mathbf{z}_k | \mathbf{x}_k) \cdot p(\mathbf{x}_k | \mathbf{Z}_{k-1})$$

$$\propto p(\mathbf{z}_k | \mathbf{x}_k) \cdot \int p(\mathbf{x}_k, \mathbf{x}_{k-1} | \mathbf{Z}_{k-1}) d\mathbf{x}_{k-1}$$

$$\propto p(\mathbf{z}_k | \mathbf{x}_k) \cdot \int p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{Z}_{k-1}) \cdot p(\mathbf{x}_{k-1} | \mathbf{Z}_{k-1}) d\mathbf{x}_{k-1}$$

$$\propto p(\mathbf{z}_k | \mathbf{x}_k) \cdot \int p(\mathbf{x}_k | \mathbf{x}_{k-1}) \cdot p(\mathbf{x}_{k-1} | \mathbf{Z}_{k-1}) d\mathbf{x}_{k-1}$$

Bayes rule:

$$p(a|b) = p(b|a)p(a)/p(b)$$

Assumption:

$$p(\mathbf{z}_k | \mathbf{x}_k, \mathbf{Z}_{k-1}) = p(\mathbf{z}_k | \mathbf{x}_k)$$

Marginalization:

$$p(a) = \int p(a, b) db$$

Factorization

from the graphical model

Assumption:

$$p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{Z}_{k-1}) = p(\mathbf{x}_k | \mathbf{x}_{k-1})$$

[Stefan Roth]

Bayesian formulation

$$p(\mathbf{x}_k | \mathbf{Z}_k) = \kappa \cdot p(z_k | \mathbf{x}_k) \cdot \int p(\mathbf{x}_k | \mathbf{x}_{k-1}) \cdot p(\mathbf{x}_{k-1} | \mathbf{Z}_{k-1}) d\mathbf{x}_{k-1}$$

$p(\mathbf{x}_k \mathbf{Z}_k)$	posterior probability at current time step
$p(z_k \mathbf{x}_k)$	likelihood
$p(\mathbf{x}_k \mathbf{x}_{k-1})$	temporal prior
$p(\mathbf{x}_{k-1} \mathbf{Z}_{k-1})$	posterior probability at previous time step
κ	normalizing term

[Stefan Roth]

Estimators

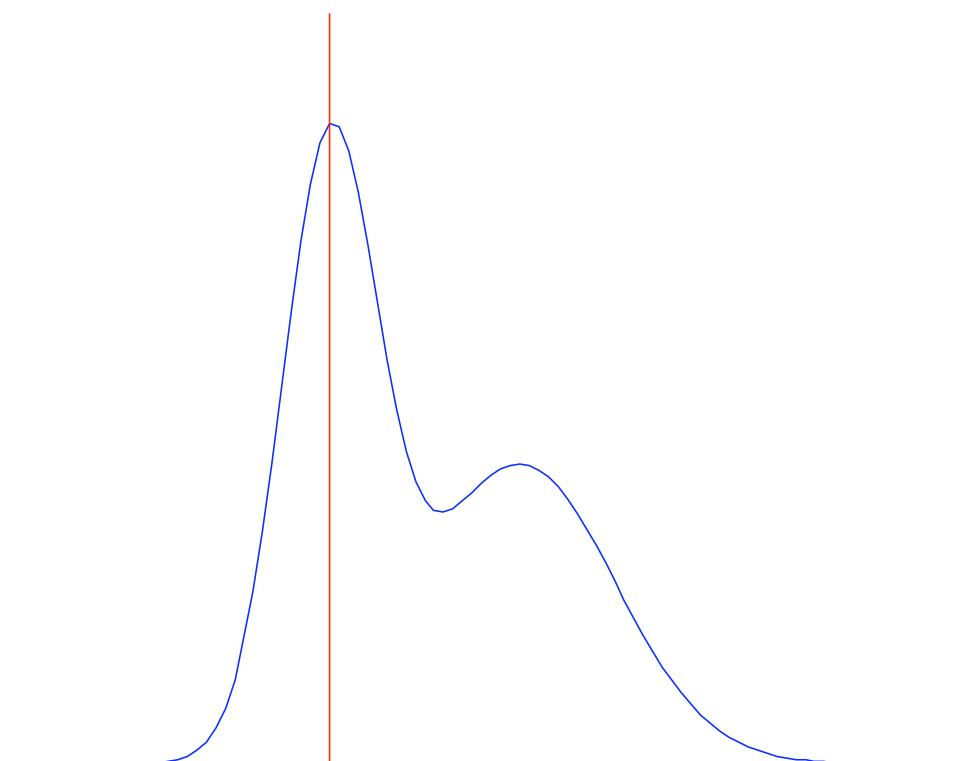
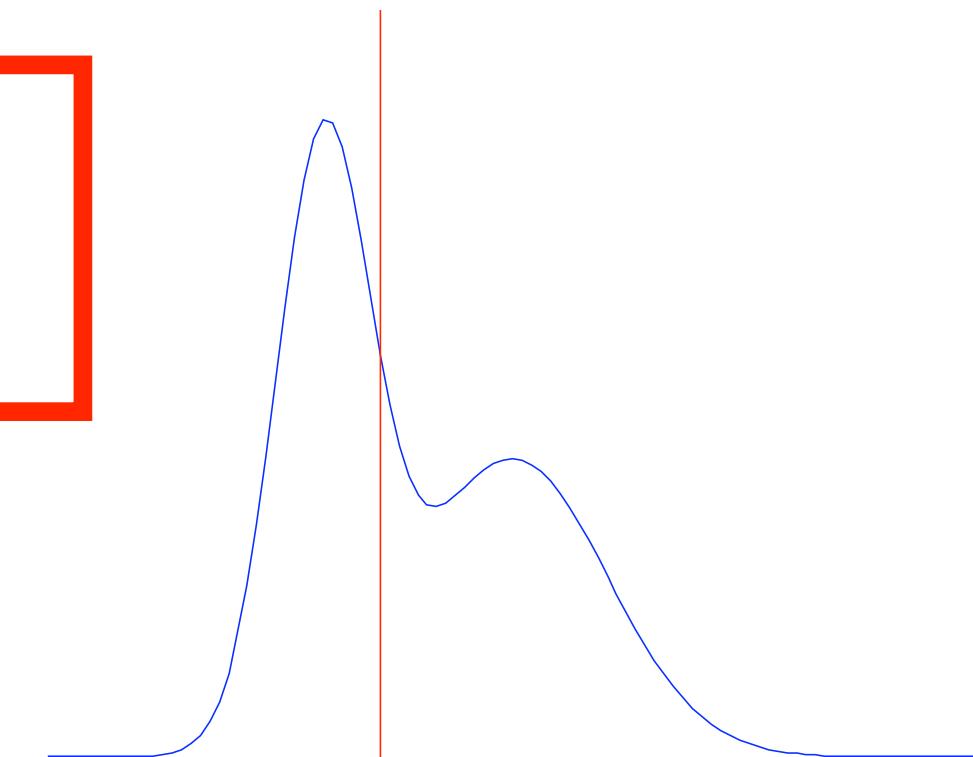
Assume the posterior probability $p(\mathbf{x}_k | \mathbf{Z}_k)$ is known:

- posterior mean

$$\hat{\mathbf{x}}_k = E(\mathbf{x}_k | \mathbf{Z}_k)$$

- maximum a posteriori (MAP)

$$\hat{\mathbf{x}}_k = \arg \max_{\mathbf{x}_k} p(\mathbf{x}_k | \mathbf{Z}_k)$$



$$p(\mathbf{x}_k | \mathbf{Z}_k)$$

[Michael Black]

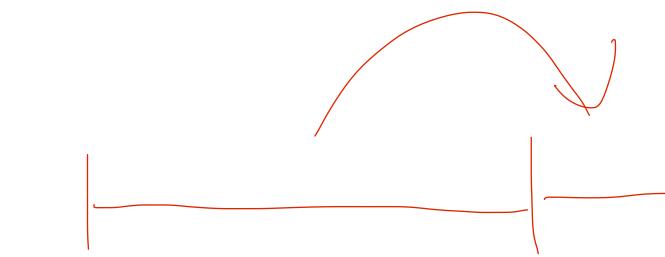
Deep networks

- It is easy to see what the networks have to output given the input
 - but it is harder (yet more useful) to understand what a network models in terms of our Bayesian formulation
- Typically, the networks are tasked to produce MAP directly, i.e.

$$\hat{\mathbf{x}}_k = \arg \max_{\mathbf{x}_k} p(\mathbf{x}_k | \mathbf{Z}_k)$$

without modelling the actual state distribution.

Online vs. offline tracking



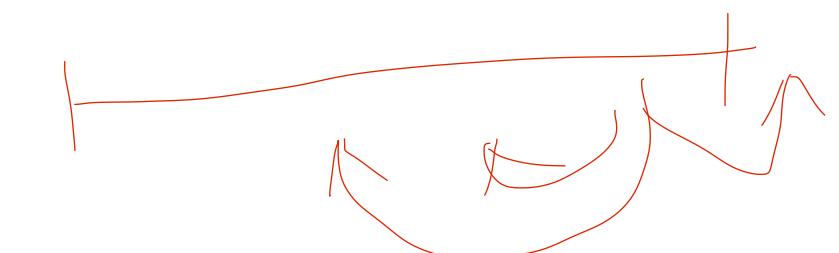
- **Online** tracking:

“given observations so far, estimate the current state”



- **Offline** tracking:

“given all observations, estimate any state”



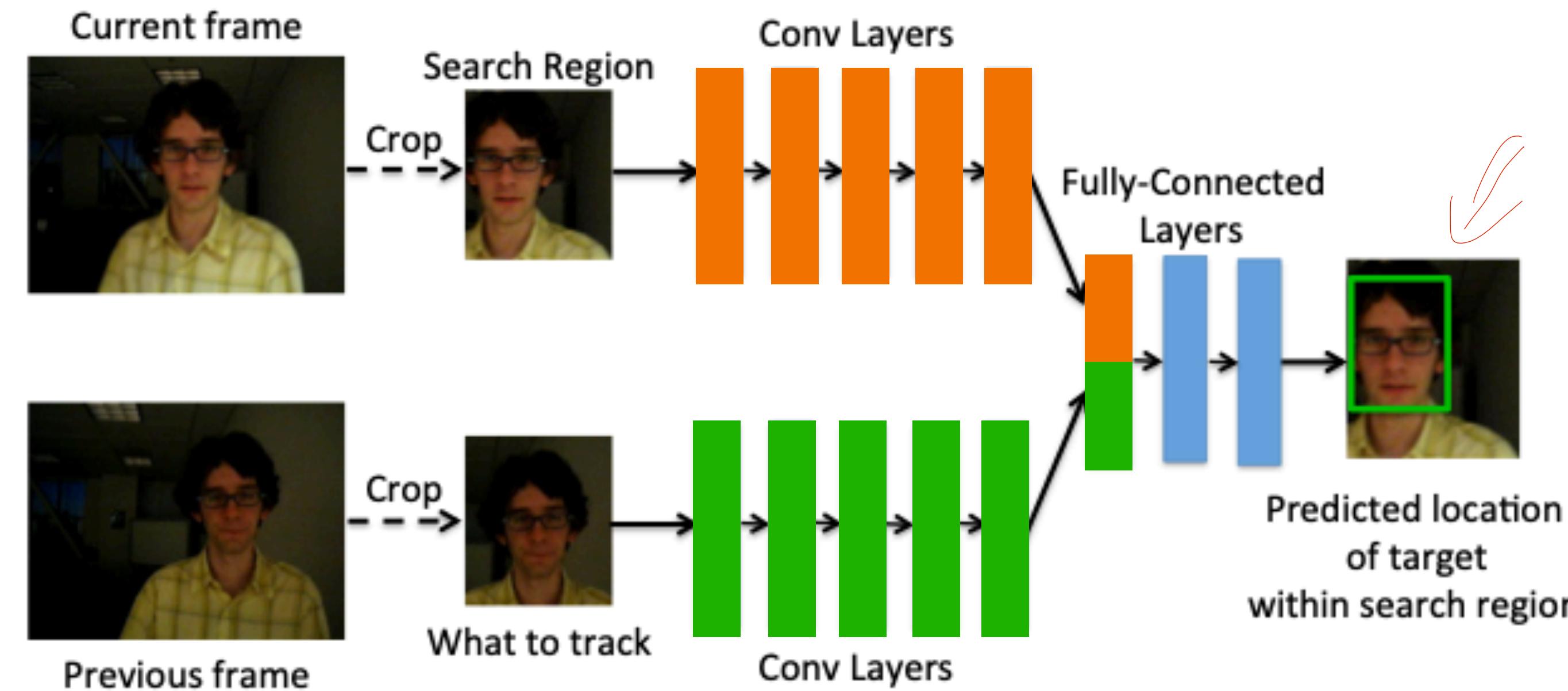
- An online tracking model can be used for offline tracking too:
 - our recursive Bayesian model will still work.

Online vs. offline tracking

- Online tracking (this lecture)
 - Processes two frames at a time
 - For real-time applications
 - Prone to drifting → hard to recover from errors or occlusions
- Offline tracking (next lecture)
 - Processes a batch of frames
 - Good to recover from occlusions (short ones as we will see)
 - Not suitable for real-time applications
 - Suitable for video analysis

Single Target Tracking (GOTURN)

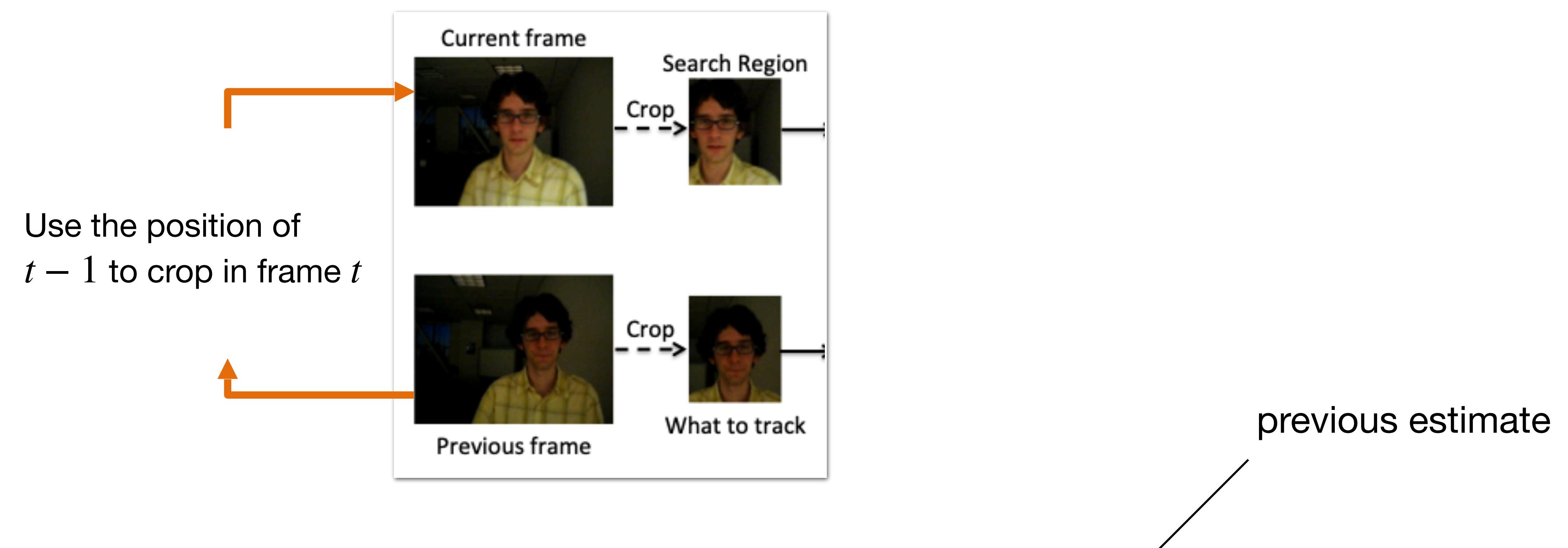
- Input: search region + template region (what to track)
- Output: bounding box coordinates in the search region



D. Held, S. Thrun, S. Savarese. "Learning to Track at 100 FPS with Deep Regression Networks". ECCV 2016.

Single Target Tracking (GOTURN)

- Quiz: Temporal prior $p(x_k | x_{k-1})$?



It's not defined analytically, but $\arg \max p(x_k | x_{k-1}) \approx \hat{x}_{k-1}$

D. Held, S. Thrun, S. Savarese. "Learning to Track at 100 FPS with Deep Regression Networks". ECCV 2016.

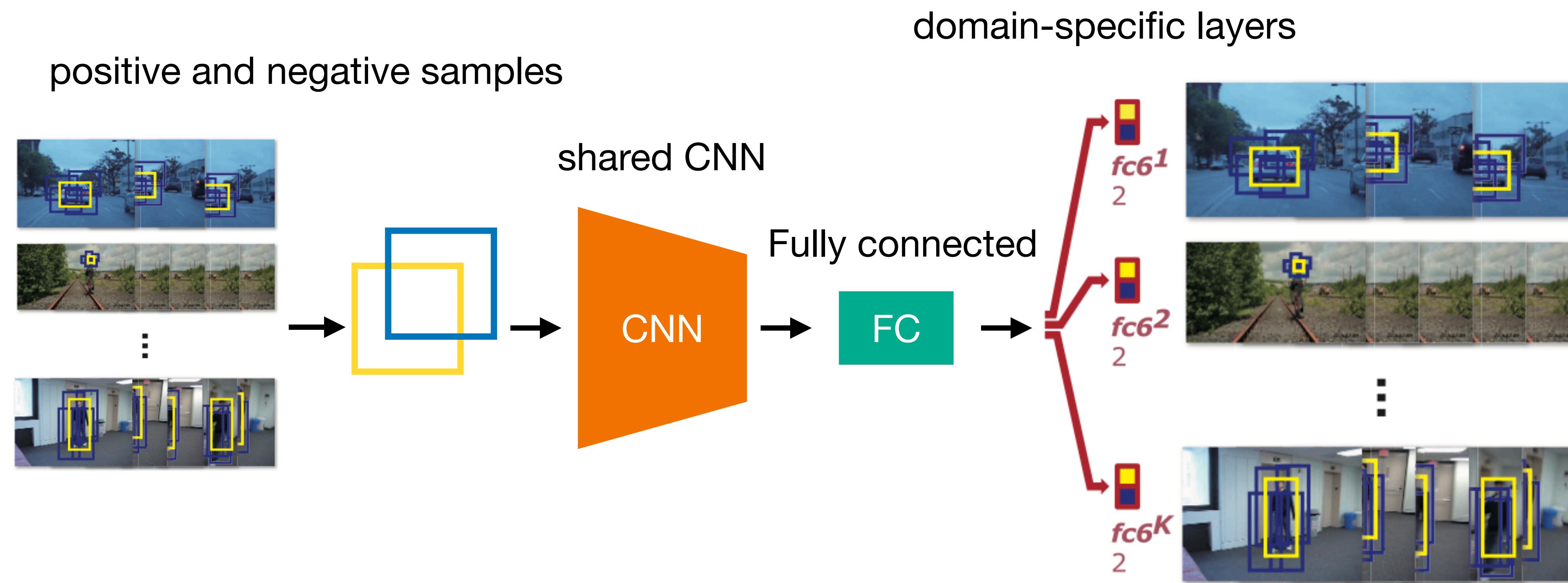
Single Target Tracking (GOTURN)

- Advantages:
 - simple: close to template matching we saw in the lectures 1-2;
 - efficient (real-time);
 - end-to-end (we can make use of large annotated data);
- Disadvantages:
 - may be sensitive to the template choice;
 - the temporal prior is too simple: fails if there is fast motion or occlusion;
 - tracking one object only.

Online adaptation

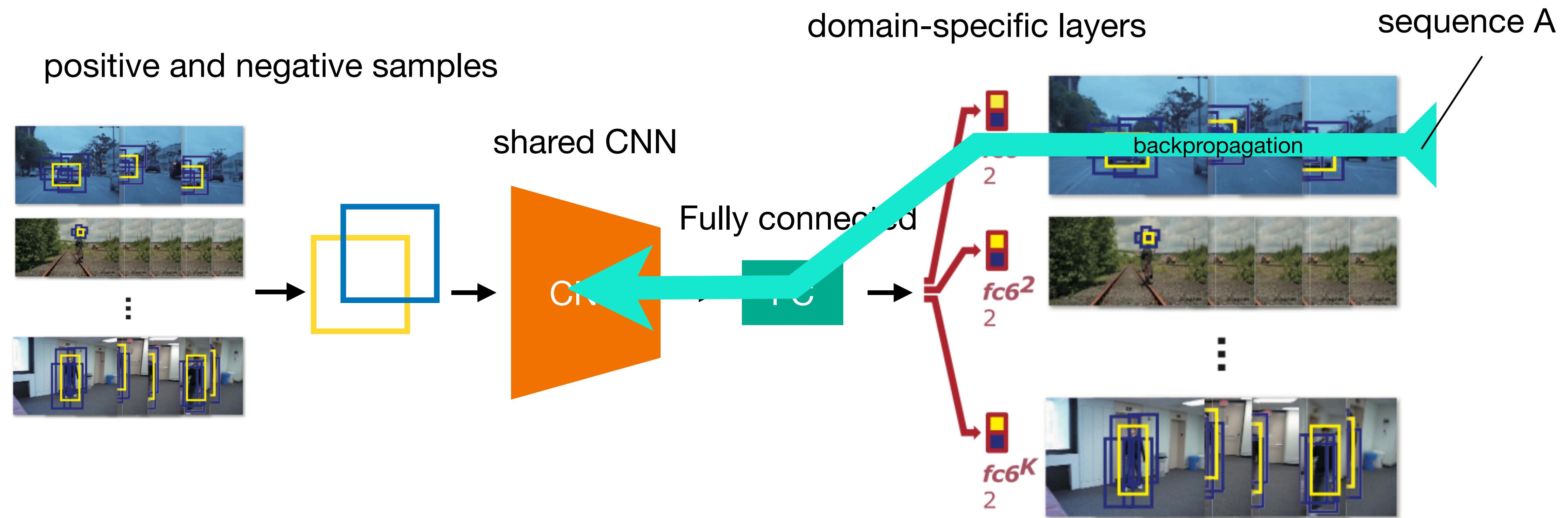
- Problem: the initial object appearance may change drastically,
 - e.g. due to occlusion, pose change, etc.
- Idea: adapt the appearance model on the (unlabelled) test sequence

Single Target Tracking (MDNet)



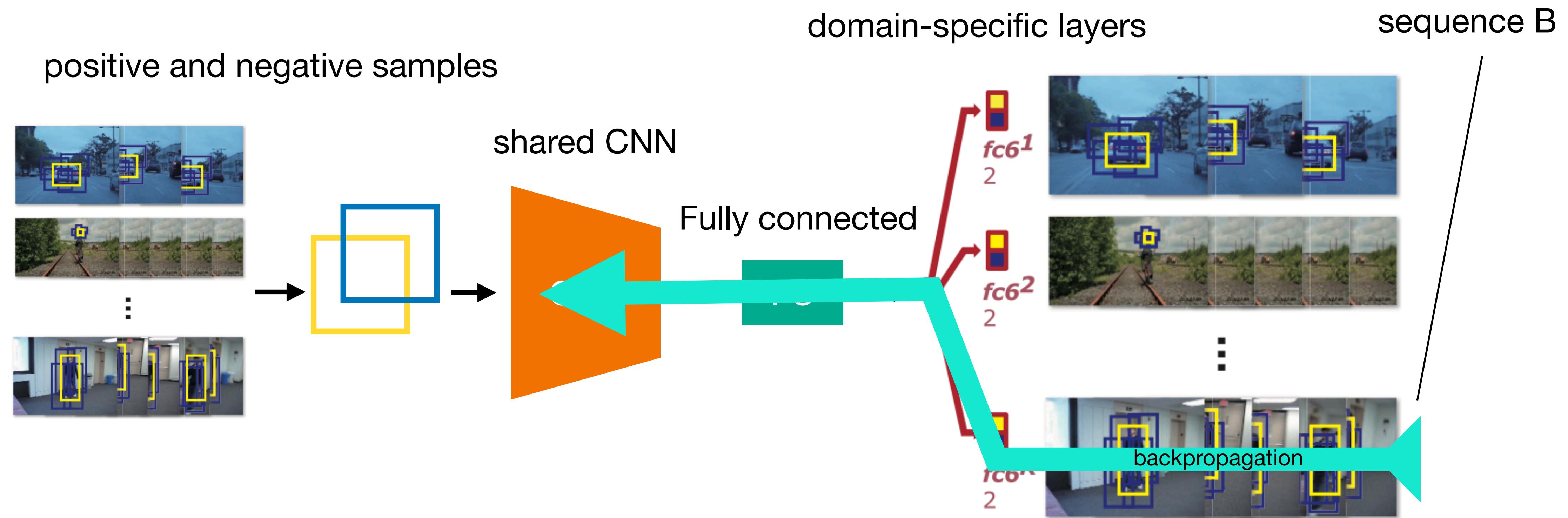
H. Nam and B. Han. „Learning Multi-Domain Convolutional Neural Networks for Visual Tracking“. CVPR 2016

Single Target Tracking (MDNet)



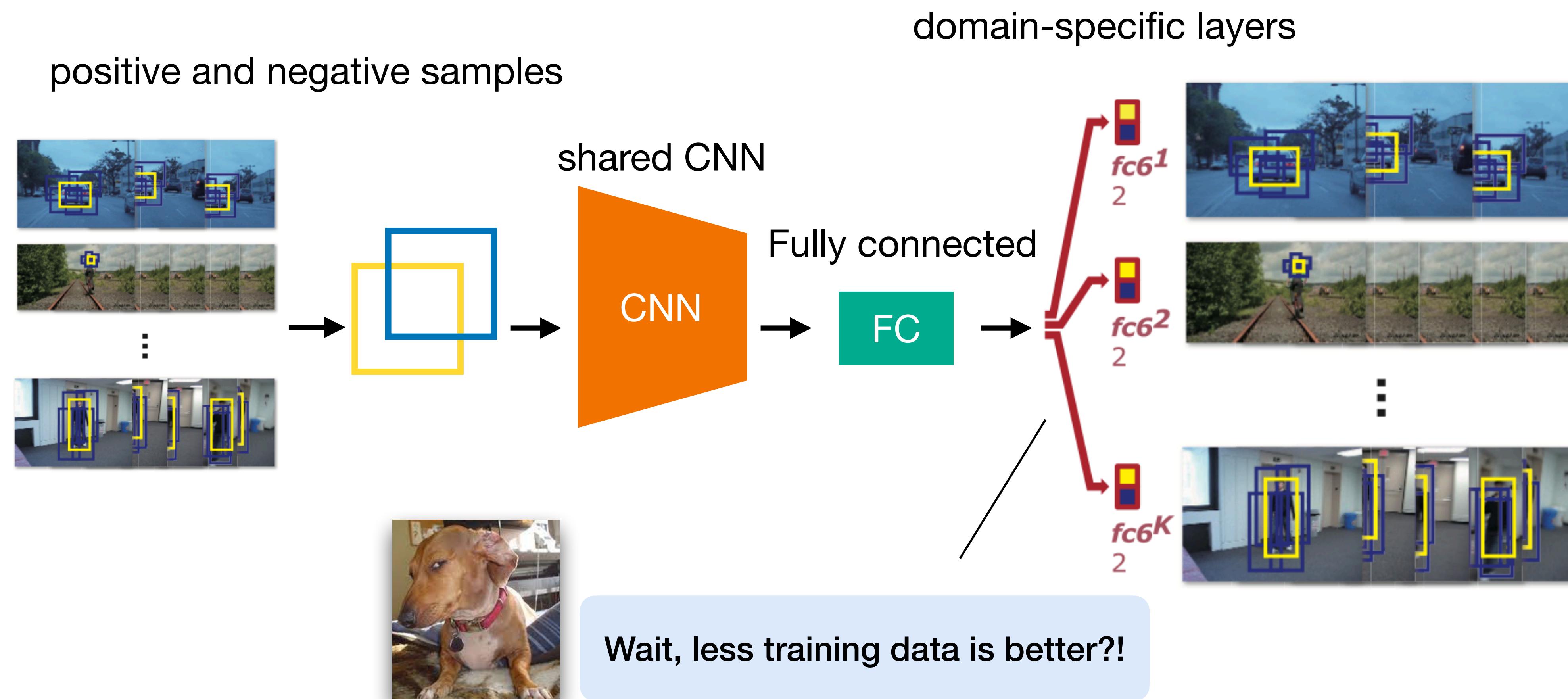
H. Nam and B. Han. „Learning Multi-Domain Convolutional Neural Networks for Visual Tracking“. CVPR 2016

Single Target Tracking (MDNet)



H. Nam and B. Han. „Learning Multi-Domain Convolutional Neural Networks for Visual Tracking“. CVPR 2016

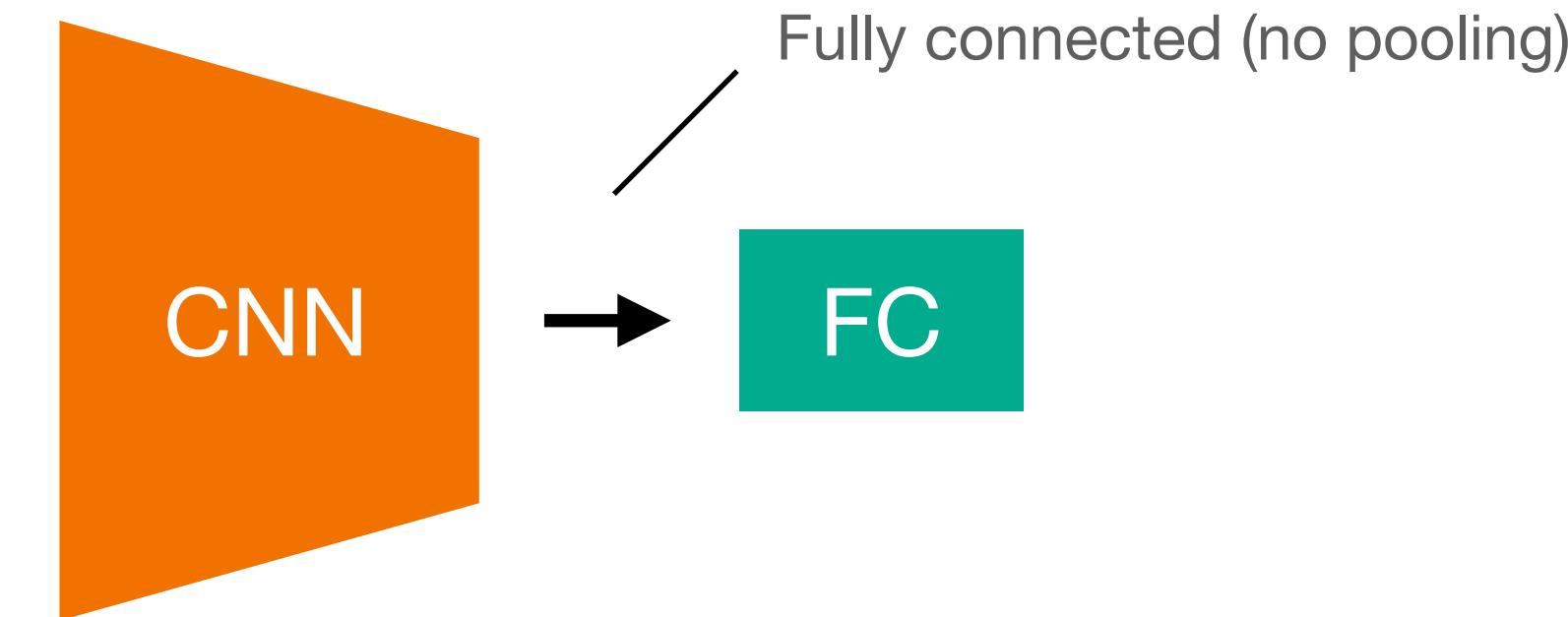
Single Target Tracking (MDNet)



H. Nam and B. Han. „Learning Multi-Domain Convolutional Neural Networks for Visual Tracking“. CVPR 2016

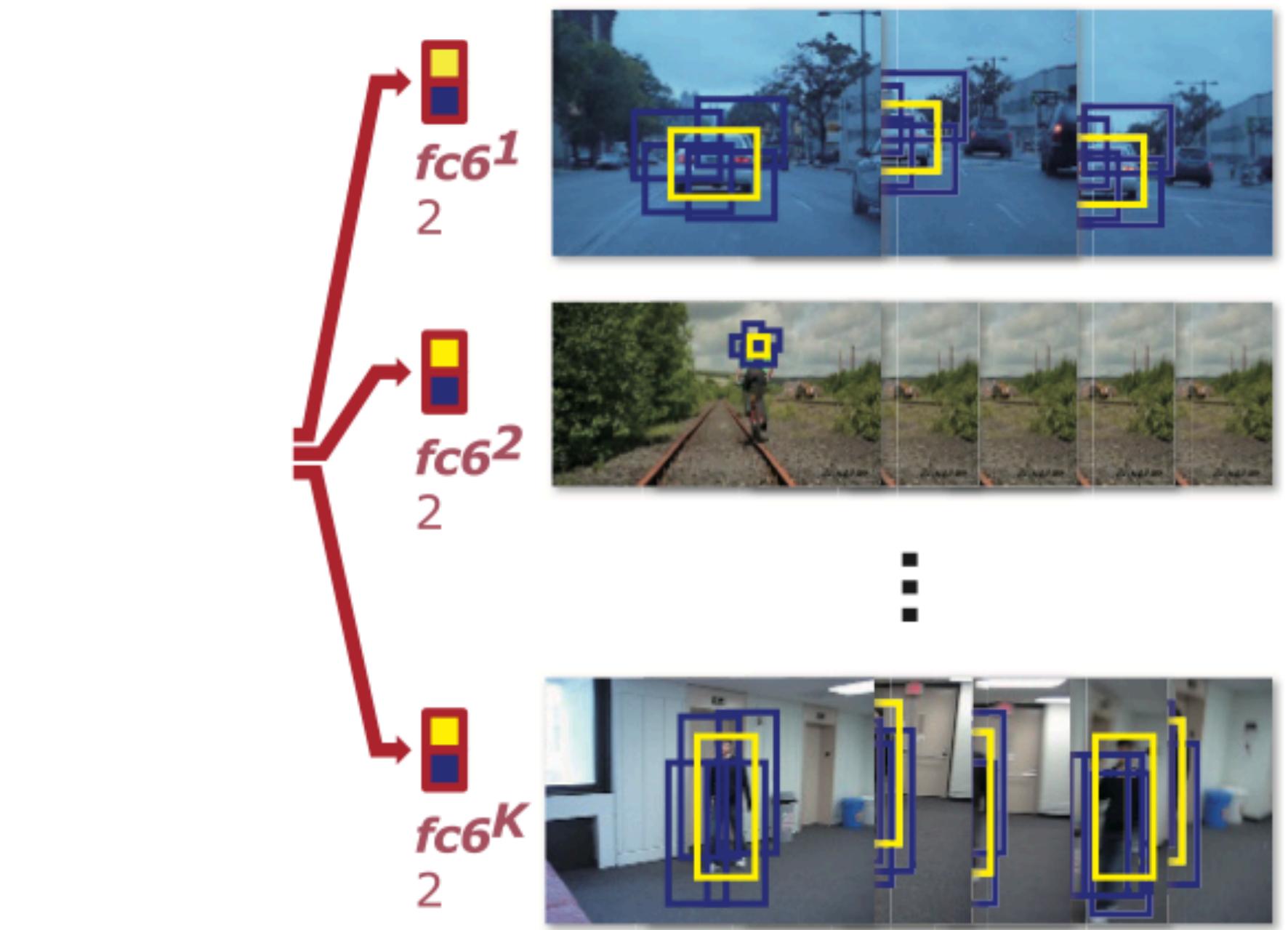
Single Target Tracking (MDNet)

- Why domain-specific layers? (Quiz)



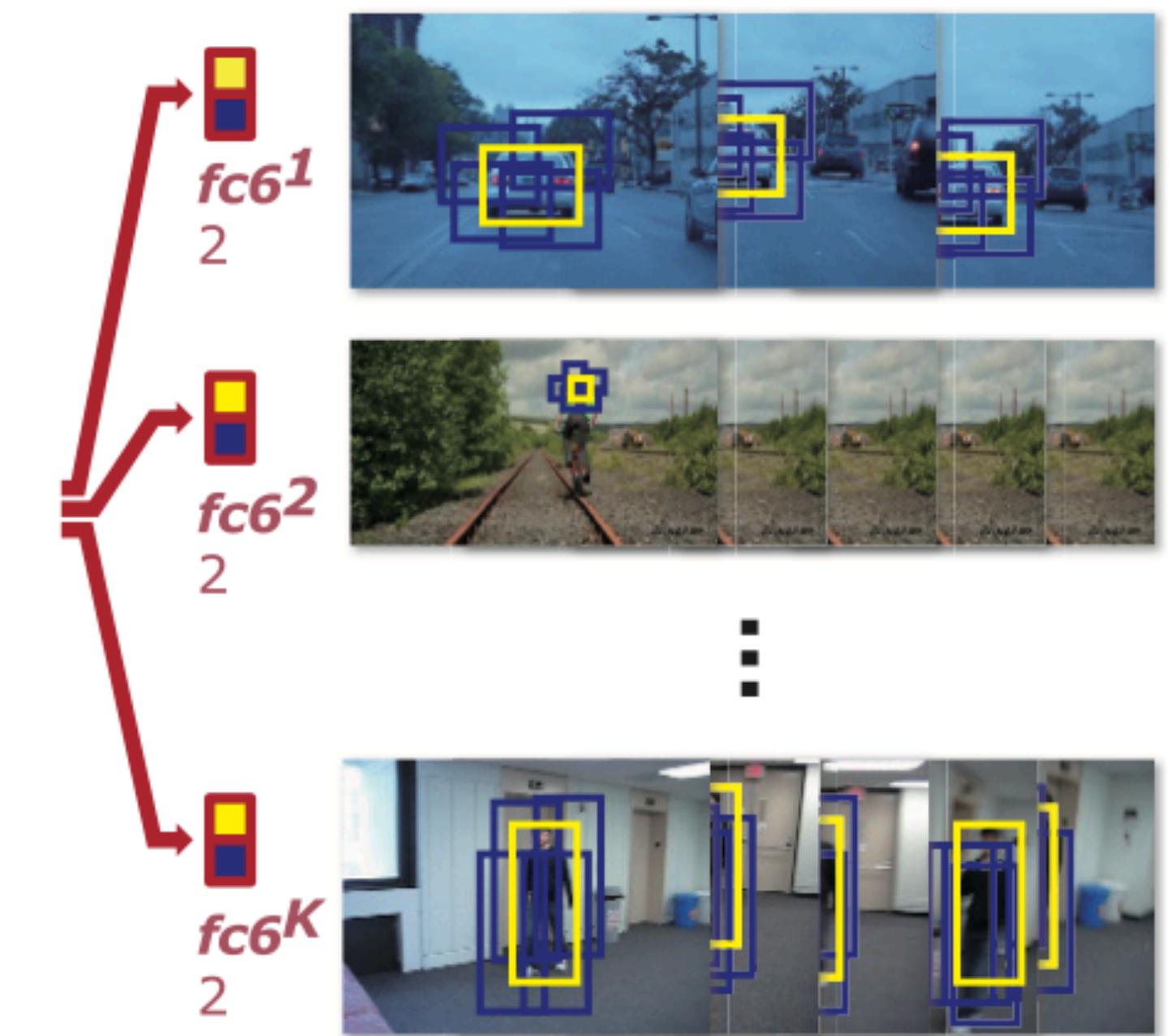
- Hint 1:

- Hint 2: Are we actually solving the same task in each sequence?

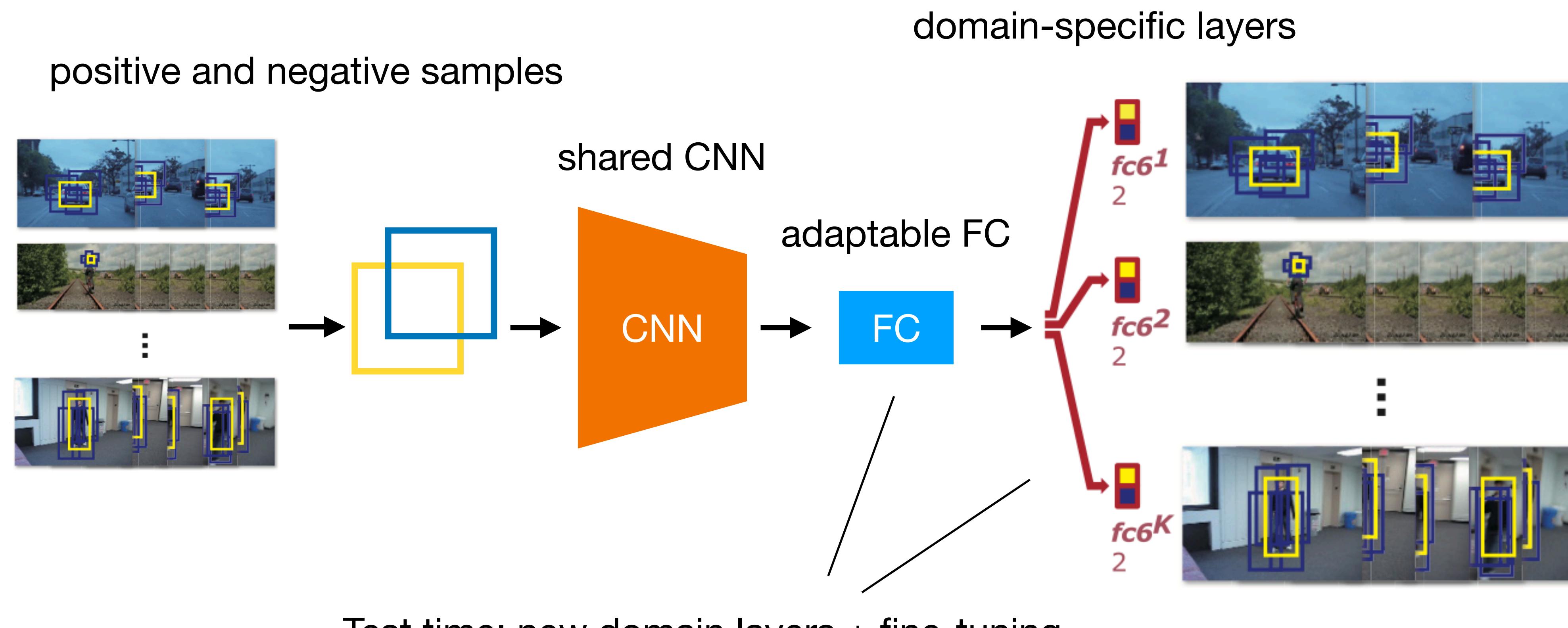


Single Target Tracking (MDNet)

- Why domain-specific layers?
 - There is no consistent (feature) pattern of the object to track.
dog / people / car
 - (Quiz) Can we come up with a fix to make it work with a single FC layer?



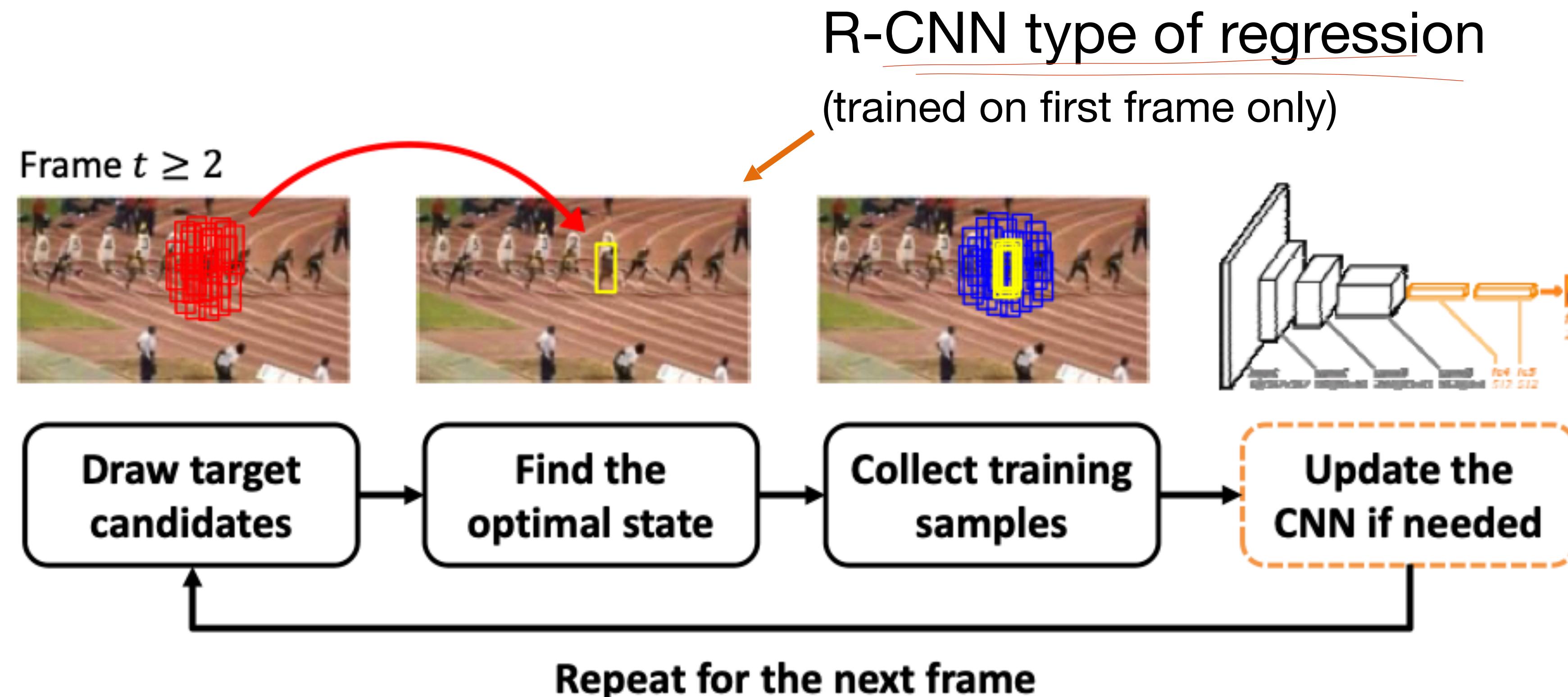
Single Target Tracking (MDNet)



H. Nam and B. Han. „Learning Multi-Domain Convolutional Neural Networks for Visual Tracking“. CVPR 2016.

Single Target Tracking (MDNet)

- Tracking with online adaptation



H. Nam and B. Han. „Learning Multi-Domain Convolutional Neural Networks for Visual Tracking“. CVPR 2016

Single Target Tracking (MDNet)

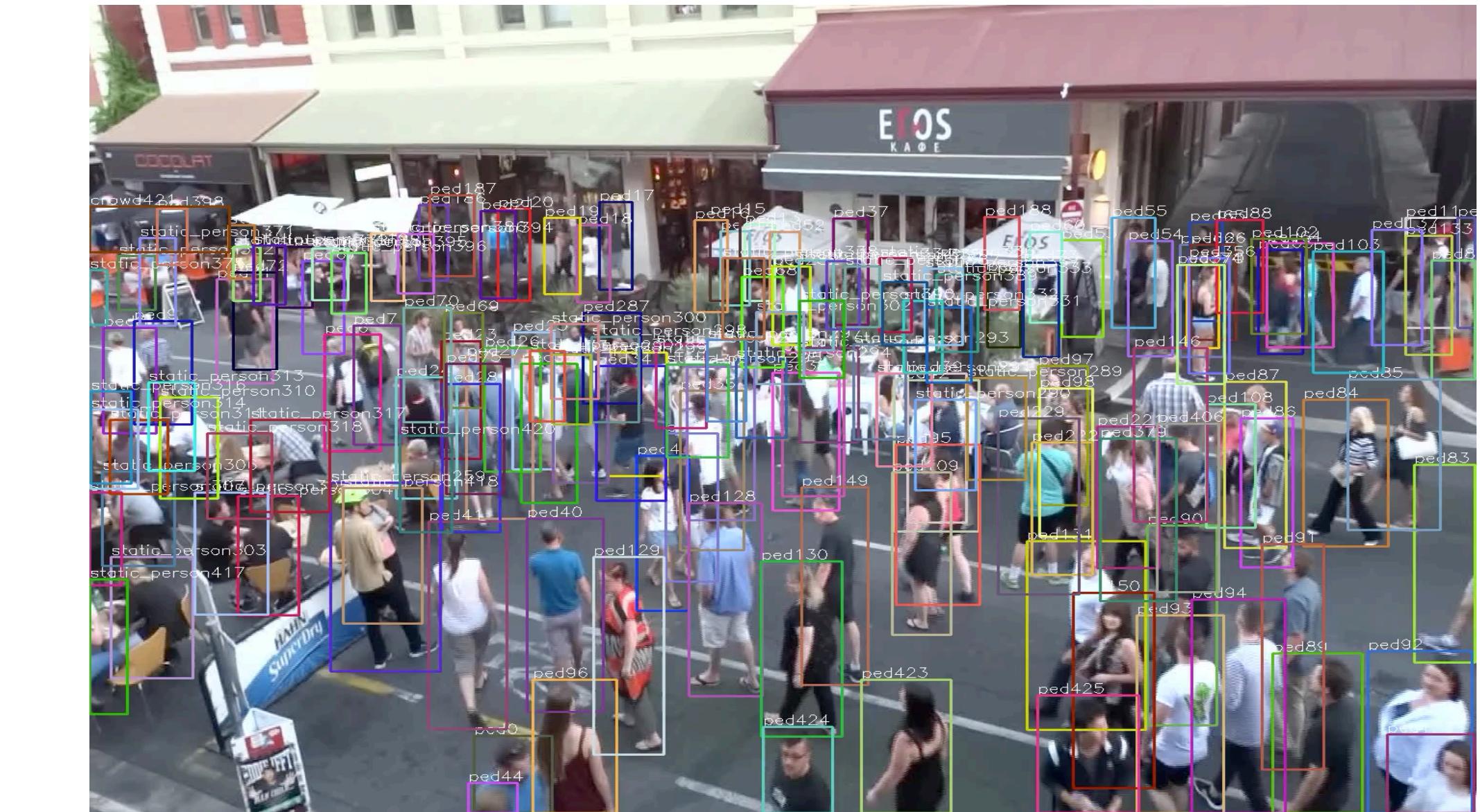
- Advantages of MDNet:
 - adaptive appearance model.
 - fine-tuning step is comparatively cheap.
 - Winner of the VOT Challenge 2015 (<http://www.votchallenge.net>)
- Disadvantages:
 - not as fast as GOTURN;
 - strong assumptions on the temporal prior.

H. Nam and B. Han. „Learning Multi-Domain Convolutional Neural Networks for Visual Tracking“. CVPR 2016

Multiple object tracking

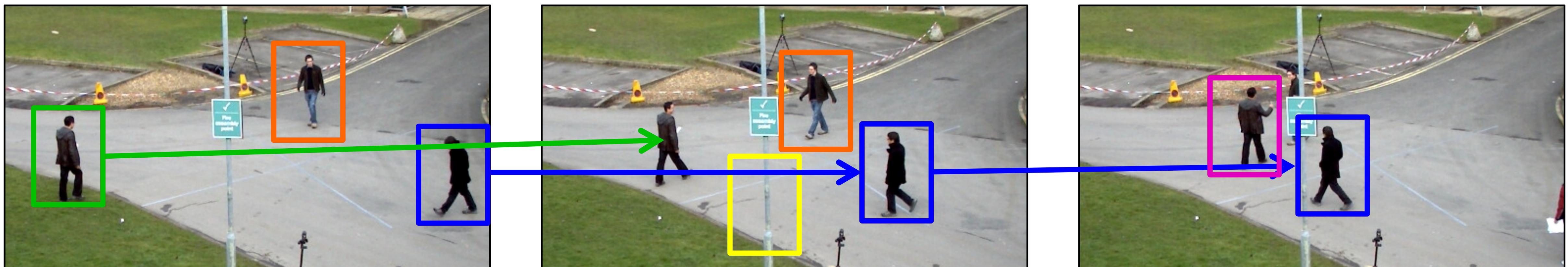
Different challenges

- Multiple objects of the same type
- Heavy occlusions
- Appearance of individual people is often very similar



Tracking by detection

- Let us assume a set of detections is provided.
- Remember, detections are not perfect!

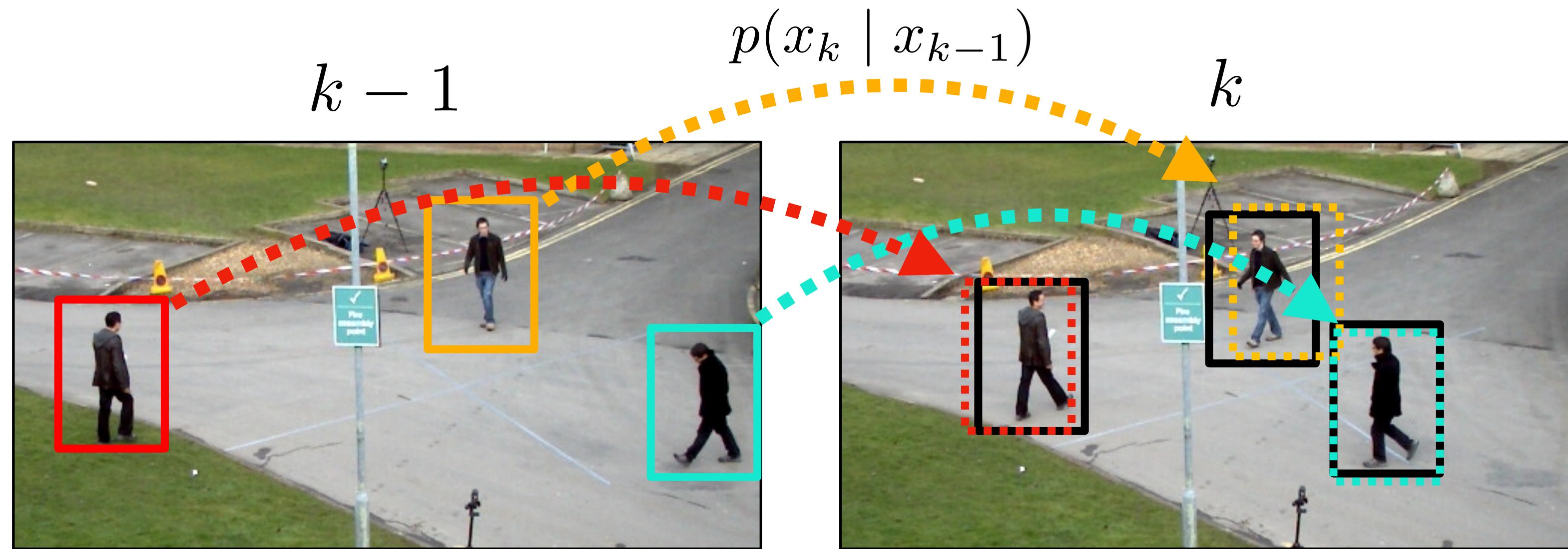


Find detections that match and form a trajectory

What is DL's role?

- 1. Track initialisation (e.g. using a detector)
 - Deep Learning → more accurate detectors.
- 2. Prediction of the next position (motion model)
 - We can learn a motion model from data!
- 3. Matching predictions with detections:
 - Learning robust metrics (robust to pose changes, partial occlusions, etc.)
 - Matching can be embedded into the model (next lecture).

Approach 1



- 1. Track initialization (e.g. using a detector)
- 2. Prediction of the next position (motion model)
- 3. **Matching** predictions with detections (appearance model)

Open questions

- How to predict the next position? (motion model)
 - Classic: Kalman filter (e.g. “state transition model”)
 - Deep learning (data-driven): LSTM/GRU (e.g. ROLO [1])

[1] Ning et al., „Spatially Supervised Recurrent Convolutional Neural Networks for Visual Object Tracking“. ISCAS 2017.

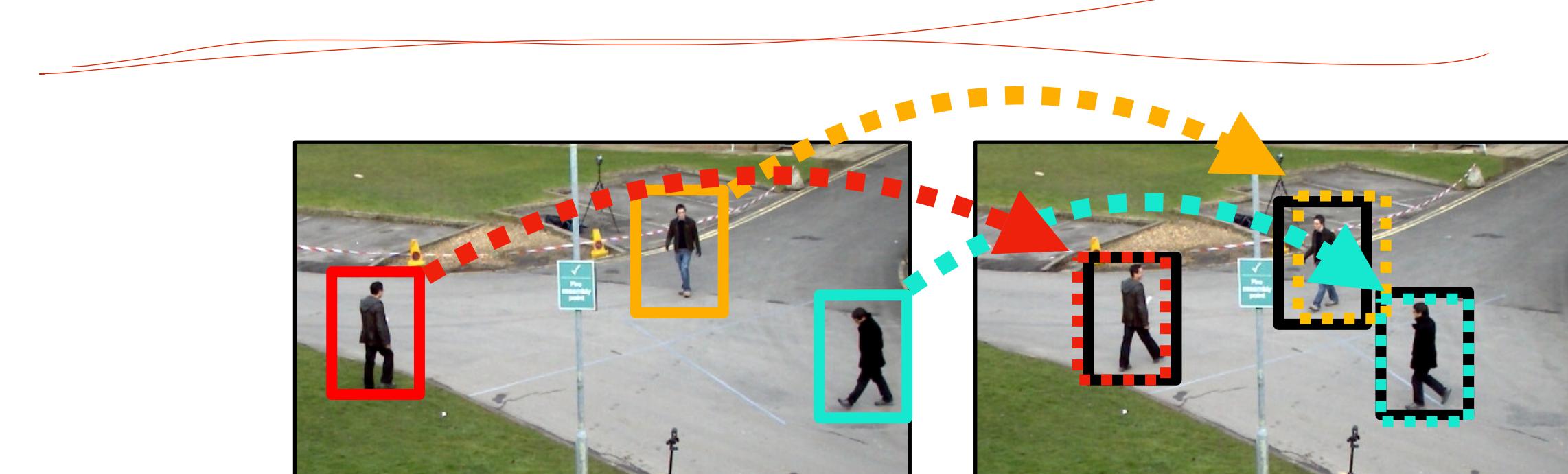
Typical models of dynamics

- Constant position:
 - i.e. no real dynamics, but if the velocity of the object is sufficiently small, this can work.
- Constant velocity (possibly unknown):
 - we assume that the velocity does not change over time.
 - as long as the object does not quickly change velocity or direction, this is a quite reasonable model.
- Constant acceleration (possibly unknown):
 - also captures the acceleration of the object.
 - this may include both the velocity, but also the directional acceleration.

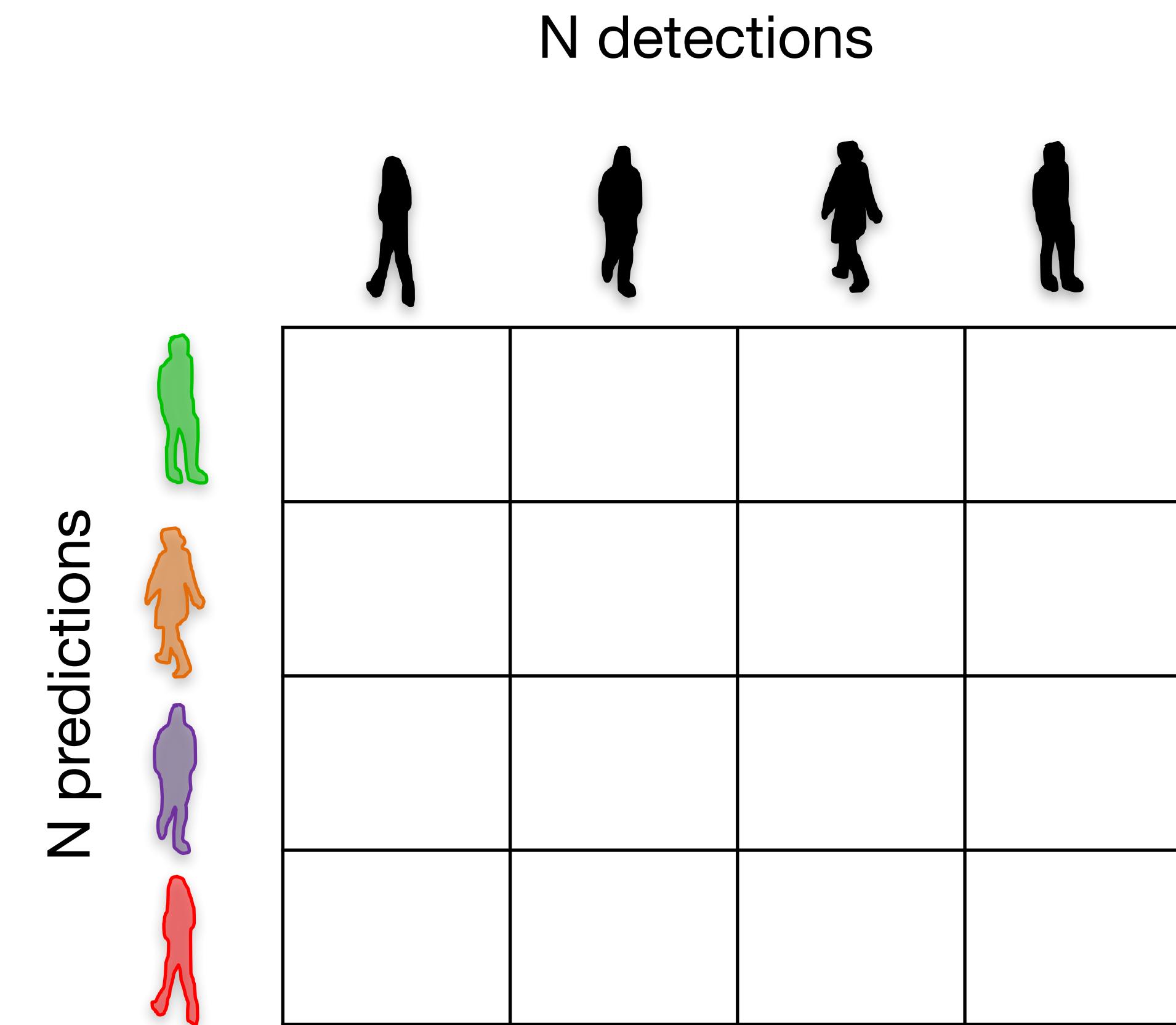
[Stefan Roth]

Open questions

- How to predict the next position? (motion model)
 - Classic: Kalman filter (e.g. “state transition model”)
 - Deep learning (data-driven): LSTM/GRU (e.g. ROLO [1])
- How to match motion prediction with detections?
 - in general: reduction to the assignment problem

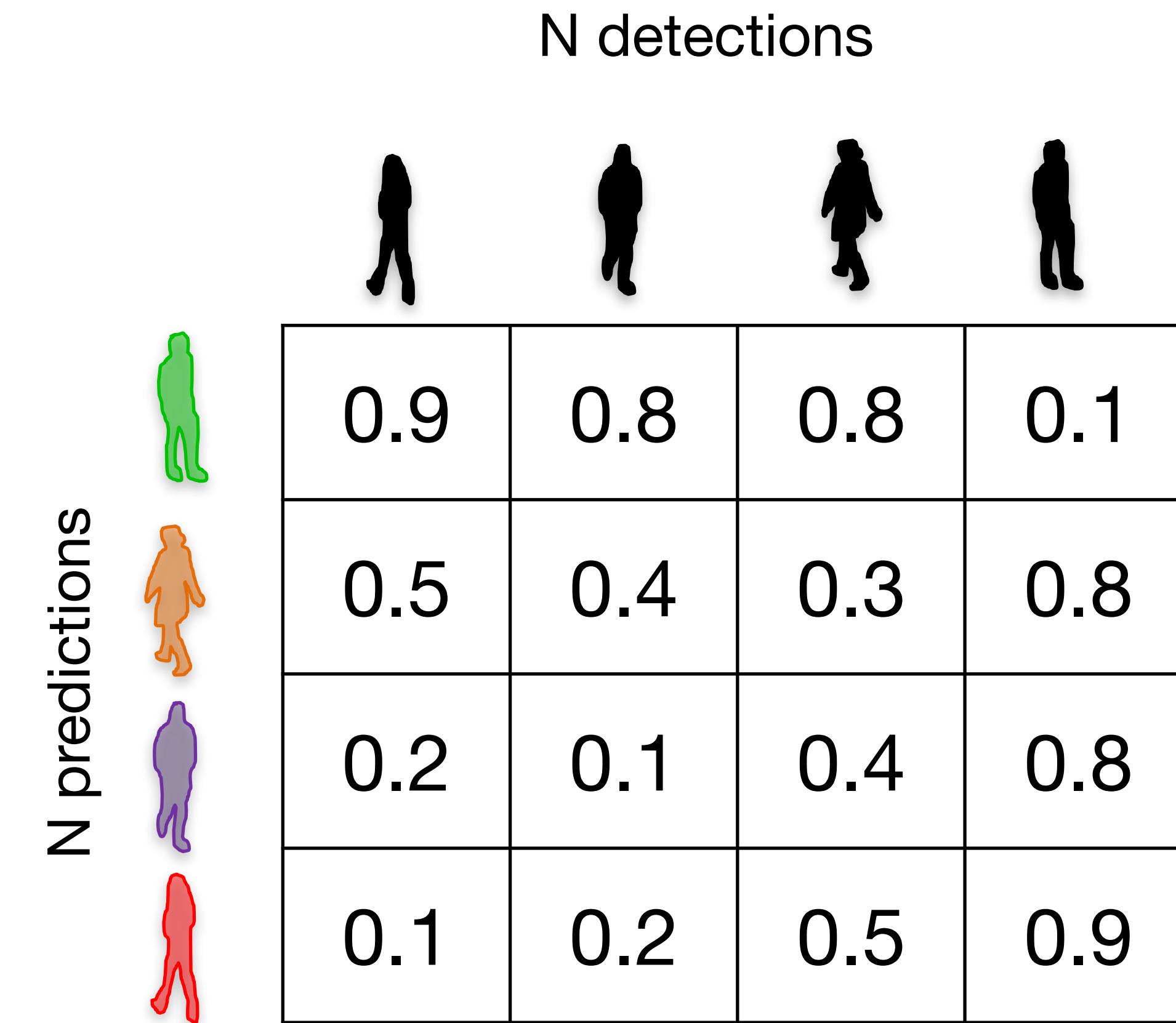


Matching predictions with detections



Bipartite matching

1. Define distances between boxes (e.g., $1 - IoU$);



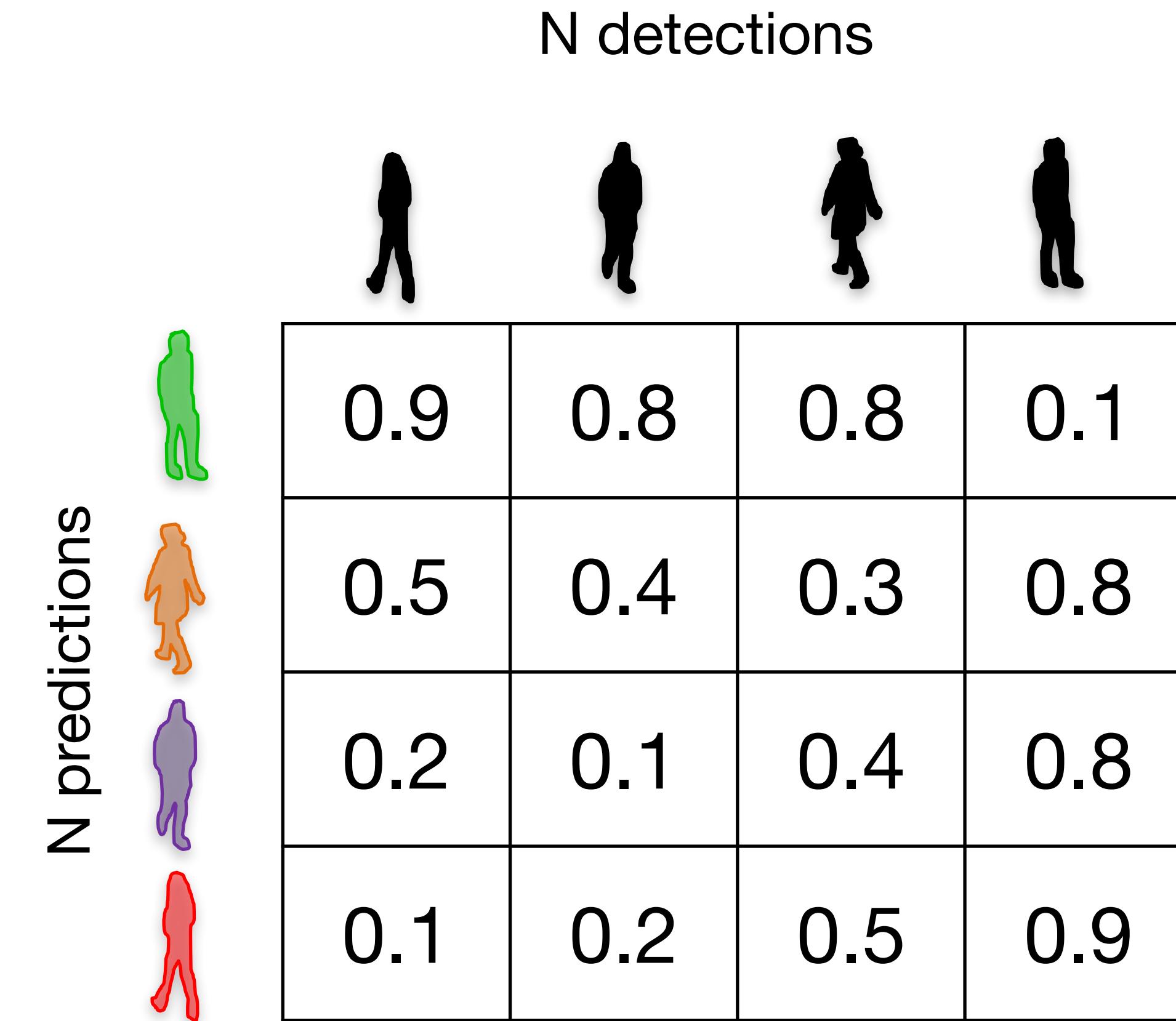
Bipartite matching

1. Define distances between boxes (e.g., $1 - IoU$);

- we obtain NxN matrix

2. Solve assignment problem

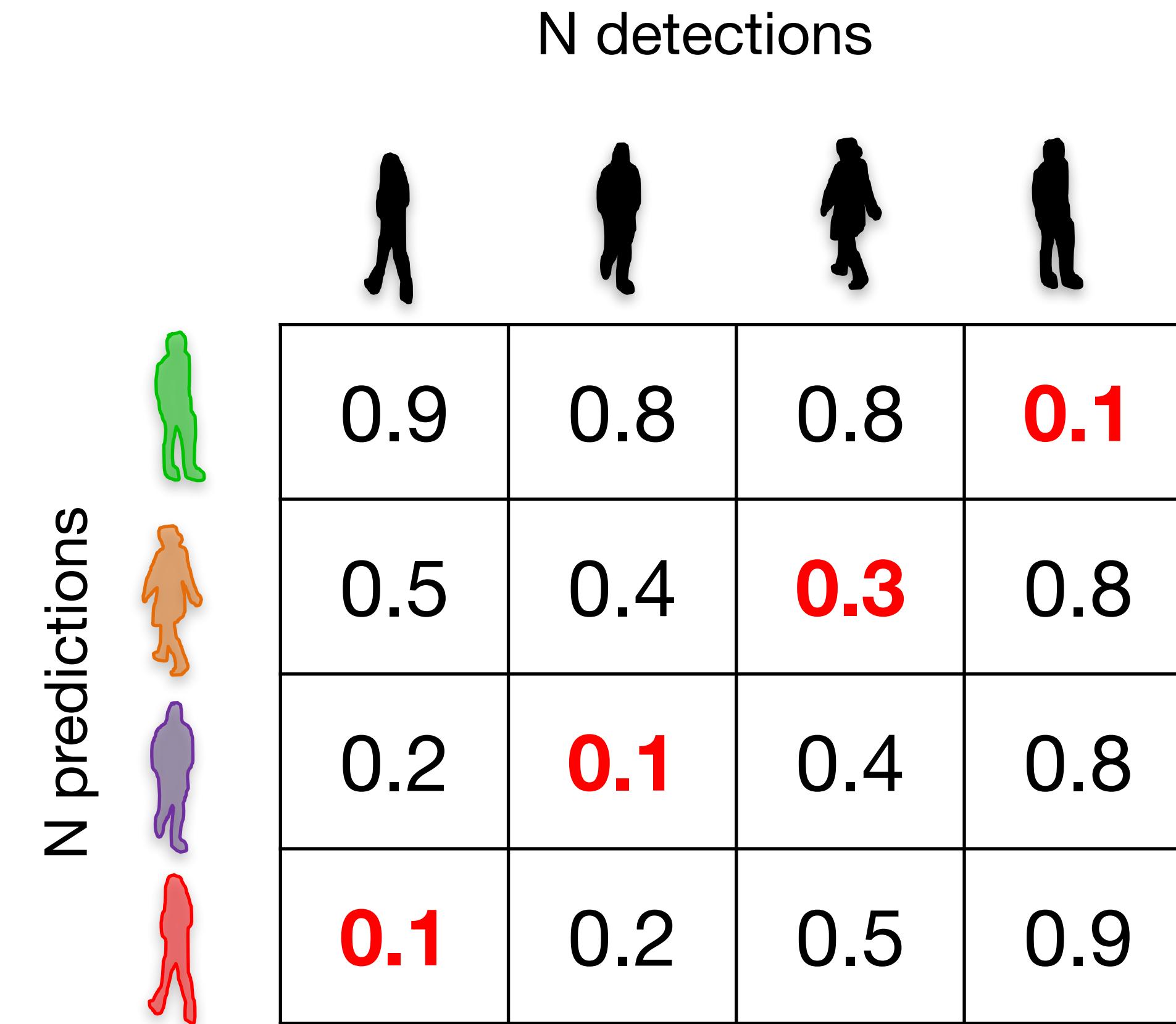
- using Hungarian algorithm*



*Demo: <http://www.hungarianalgorithm.com/solve.php>

Bipartite matching

1. Define distances between boxes (e.g., $1 - IoU$):
 - we obtain NxN matrix
2. Solve assignment problem
 - using Hungarian algorithm* $O(N^3)$
3. The bipartite matching solution corresponds to the minimum total cost



*Demo: <http://www.hungarianalgorithm.com/solve.php>

Bipartite matching

- What happens if one detection is missing?

N-1 detections



N predictions	0.9	0.8	0.8	
0.5	0.4	0.3		
0.2	0.1	0.4		
0.1	0.2	0.5		

Bipartite matching

- What happens if one detection is missing?
 - add a pseudo detection with a fixed cost (e.g. 0);

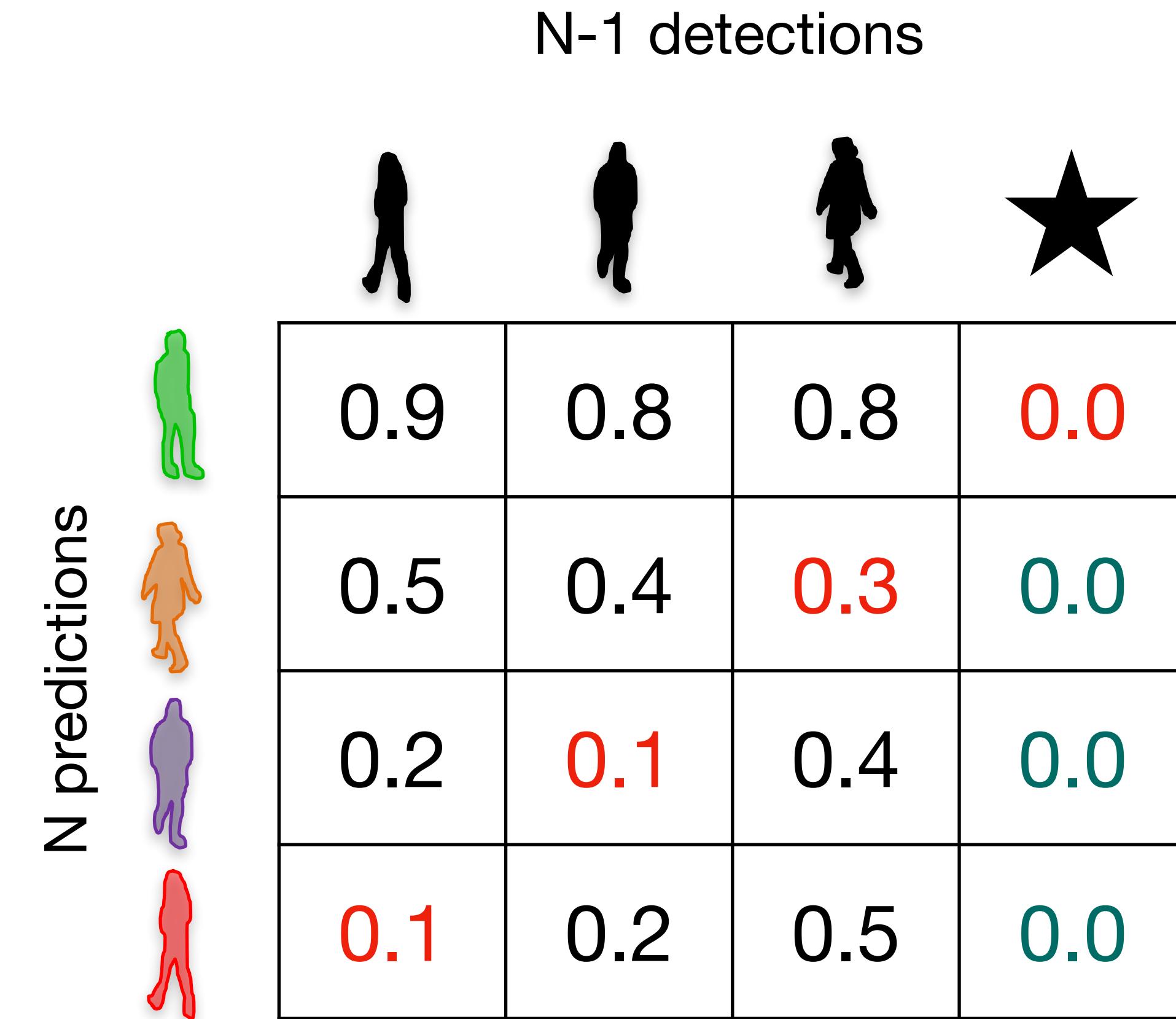

N-1 detections



N predictions	0.9	0.8	0.8	0.0
0.5	0.4	0.3	0.0	
0.2	0.1	0.4	0.0	
0.1	0.2	0.5	0.0	

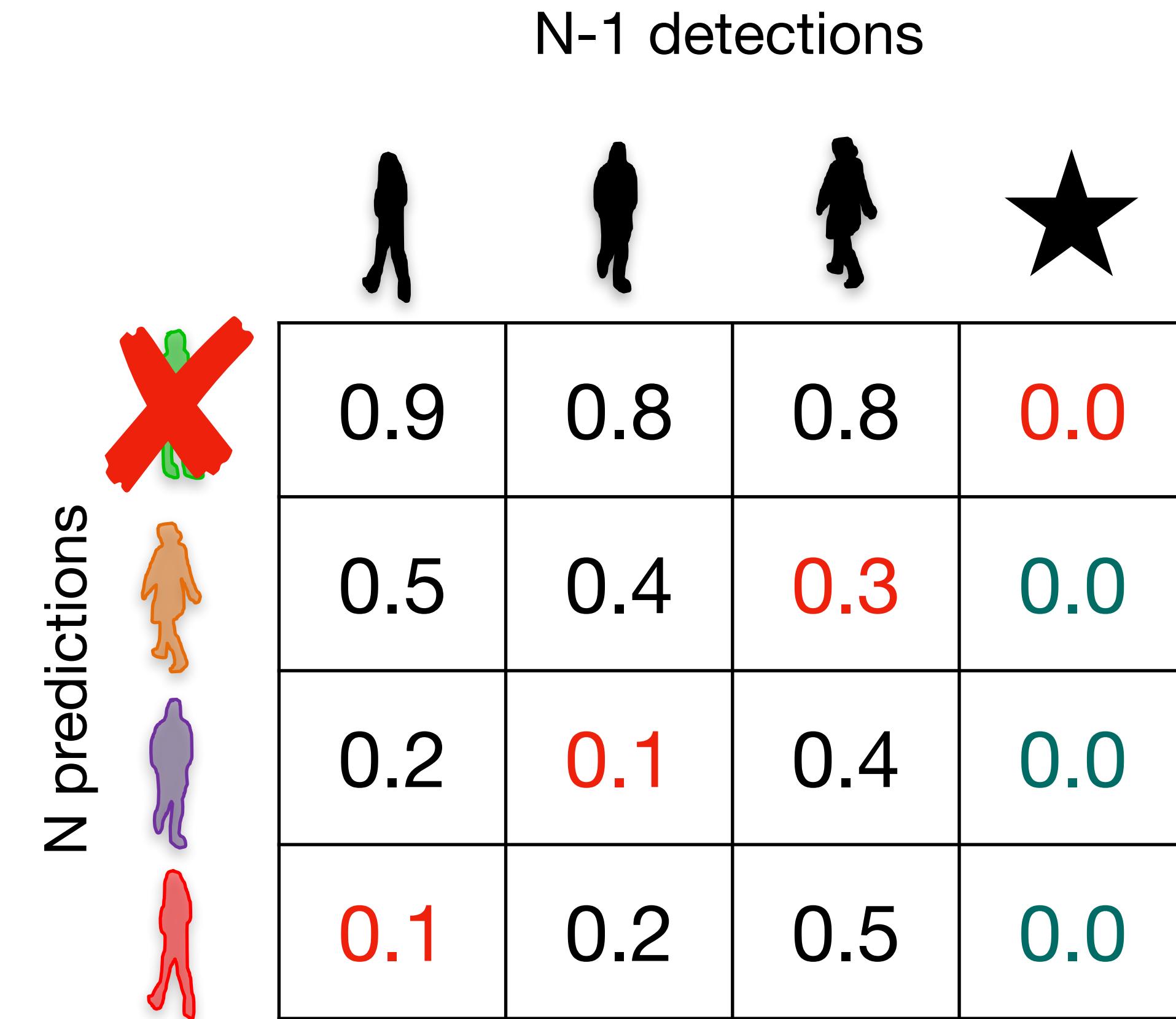
Bipartite matching

- What happens if one detection is missing?
 - add a pseudo detection with a fixed cost (e.g. 0);
 - run the Hungarian method as before;



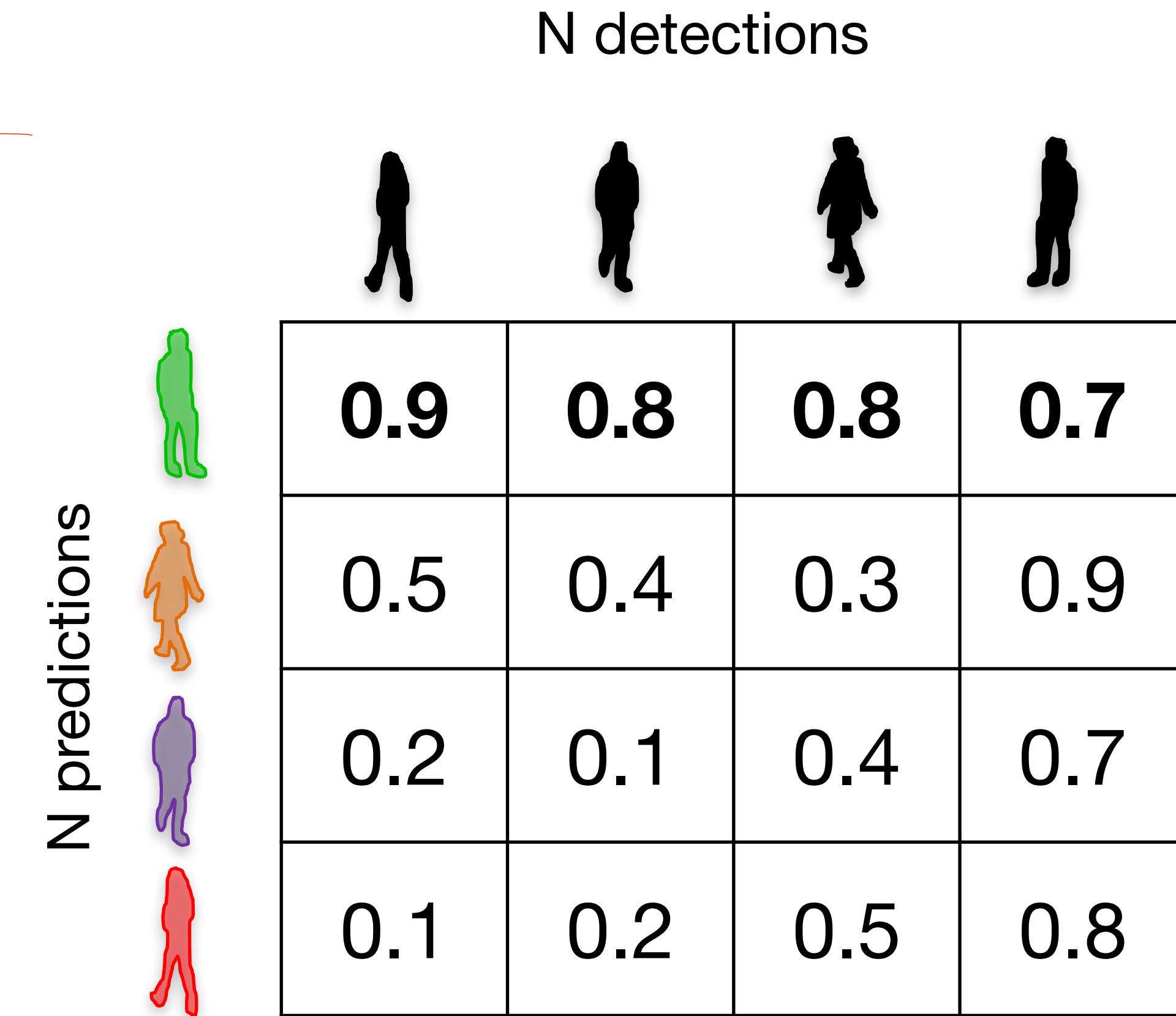
Bipartite matching

- What happens if one detection is missing?
 - add a pseudo detection with a fixed cost (e.g. 0);
 - run the Hungarian method as before;
 - discard the assignment to the pseudo node



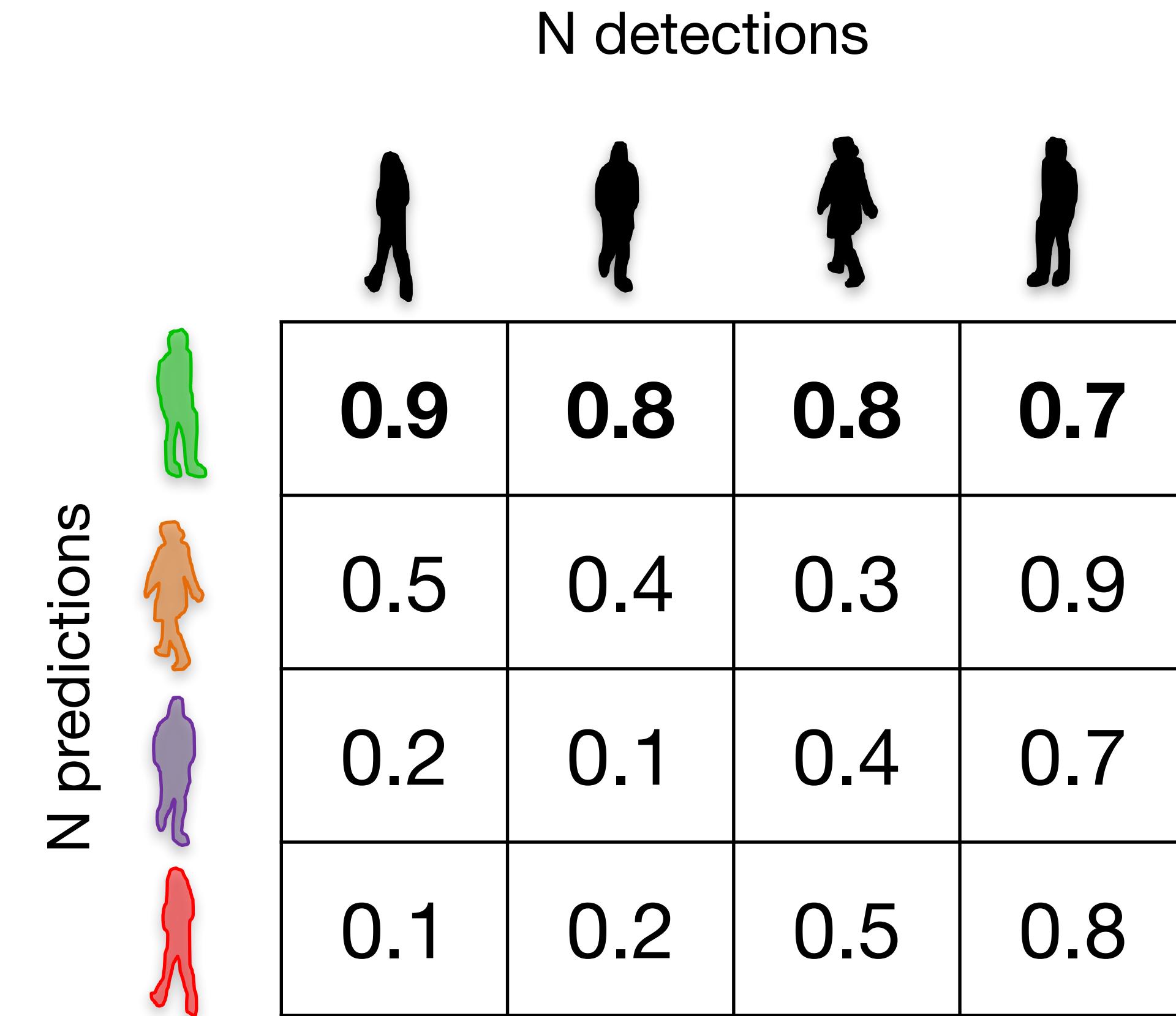
Bipartite matching

- What happens if no prediction is suitable?
 - e.g. the object leaves the frame.



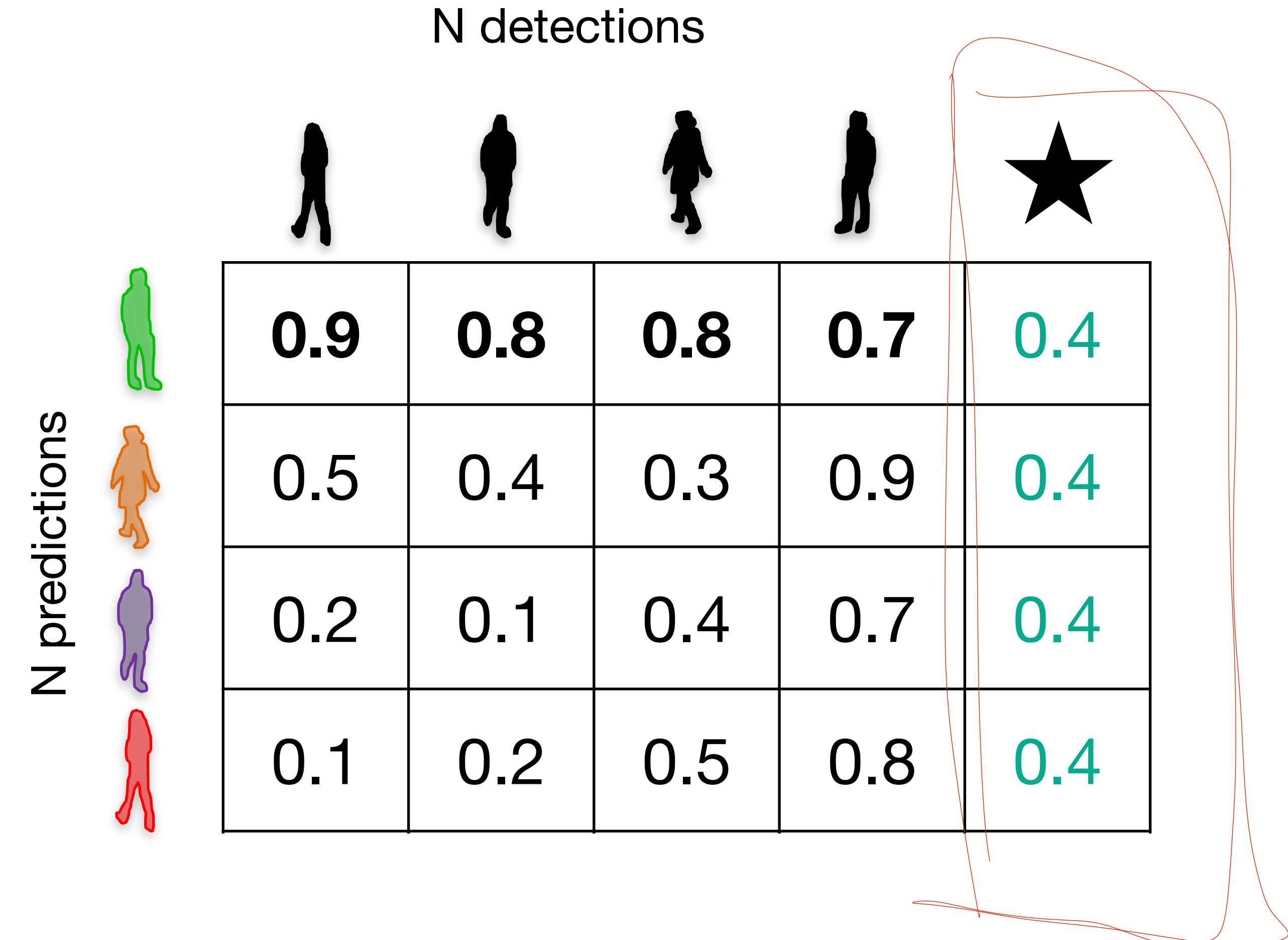
Bipartite matching

- What happens if no prediction is suitable?
 - e.g. the object leaves the frame.
- Solution: pseudo node
 - its value will define a threshold



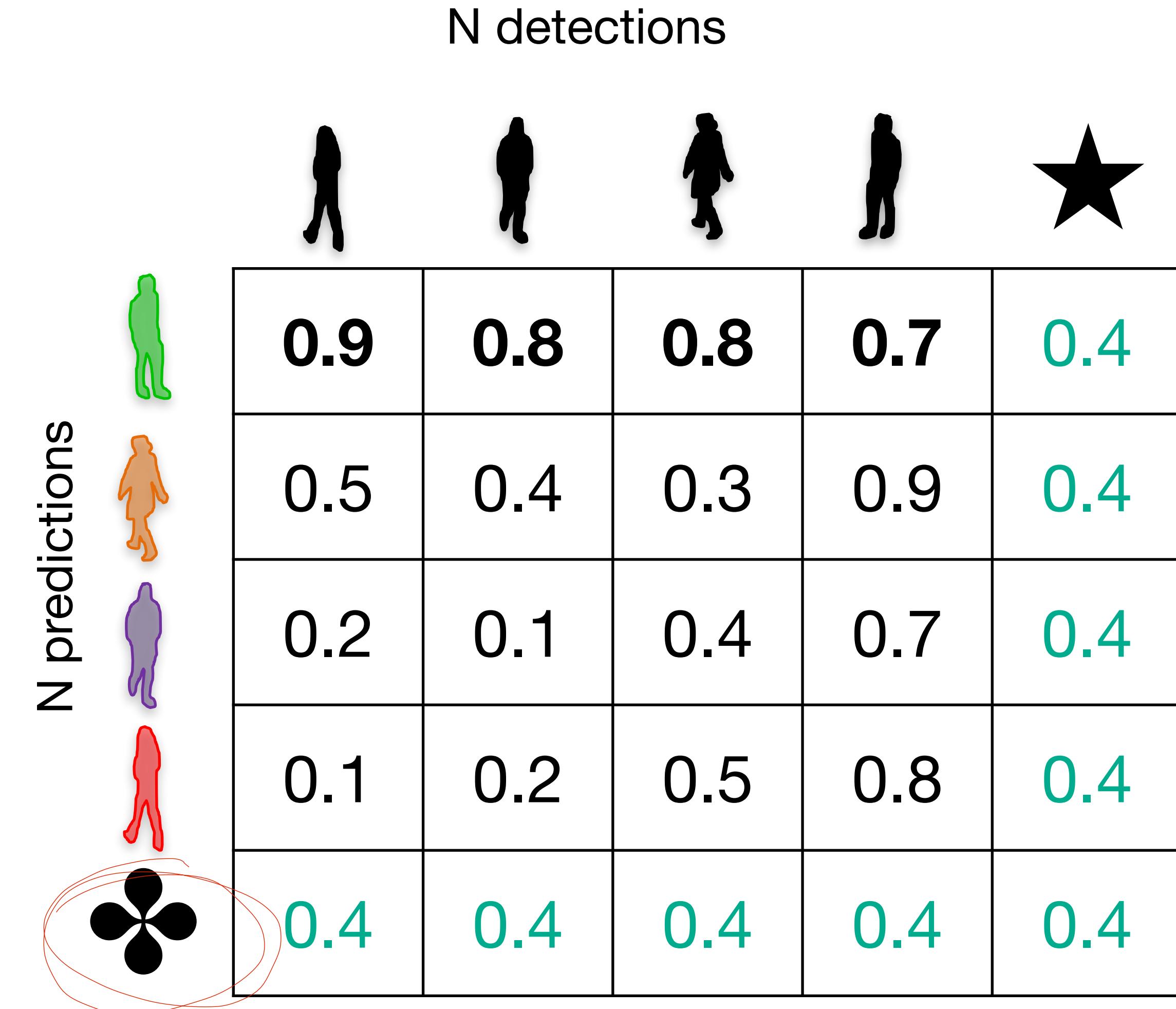
Bipartite matching

- What happens if no prediction is suitable?
 - e.g. the object leaves the frame.
- Solution: pseudo node
 - its value will define a threshold

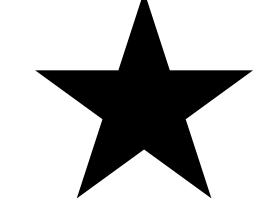
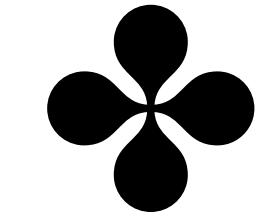


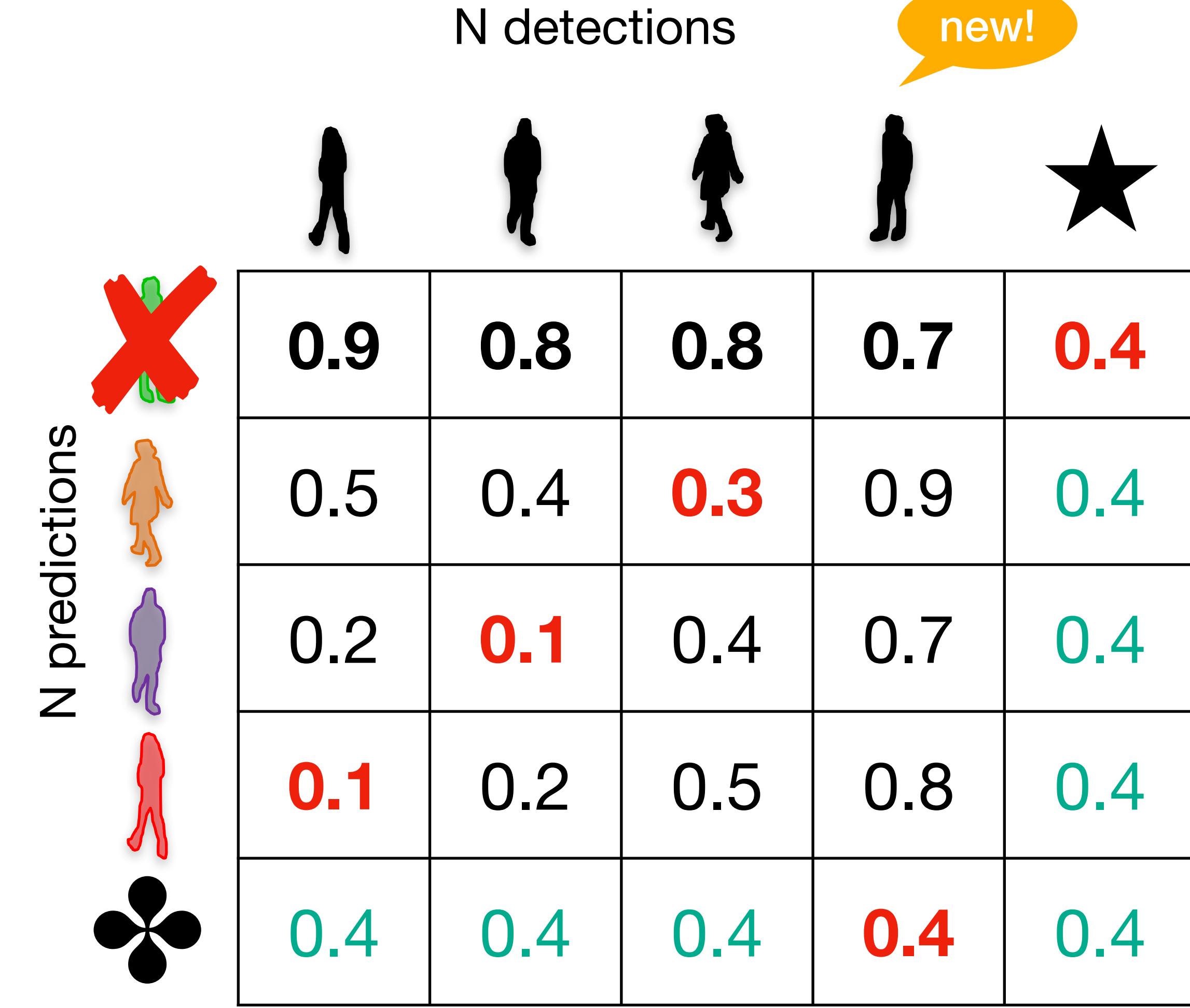
Bipartite matching

- What happens if no prediction is suitable?
 - e.g. the object leaves the frame.
- Solution: pseudo node
 - its value will define a threshold
 - we may need to balance it out



Bipartite matching

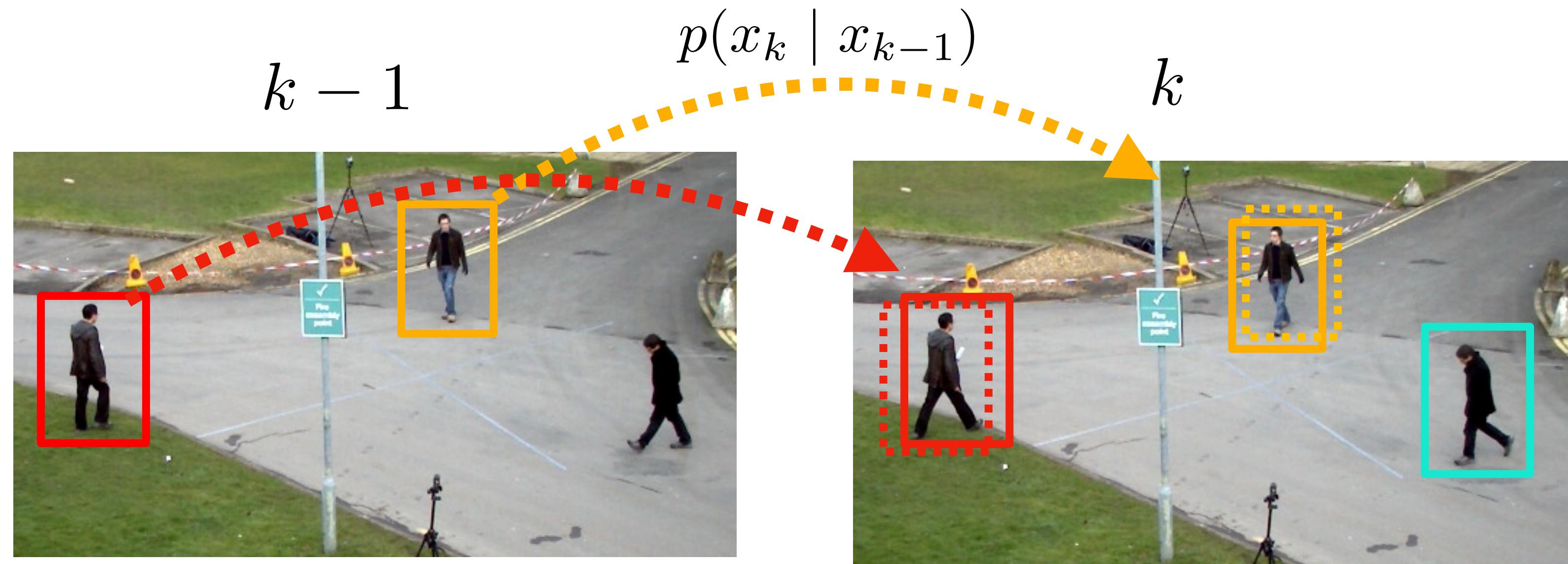
- What happens if no prediction is suitable?
 - e.g. the object leaves the frame.
- Solution: pseudo node
 - its value will define a threshold
 - we may need to balance it out
- Remove tracks: 
- New tracks: 



Open questions

- How to predict the next position? (motion model)
 - Classic: Kalman filter
 - Deep learning: LSTM/GRU (e.g. ROLO)
- How to match motion prediction with detections?
 - in general: reduction to the assignment problem;
 - **small motion: refine the boxes with a regression head**

Approach 2



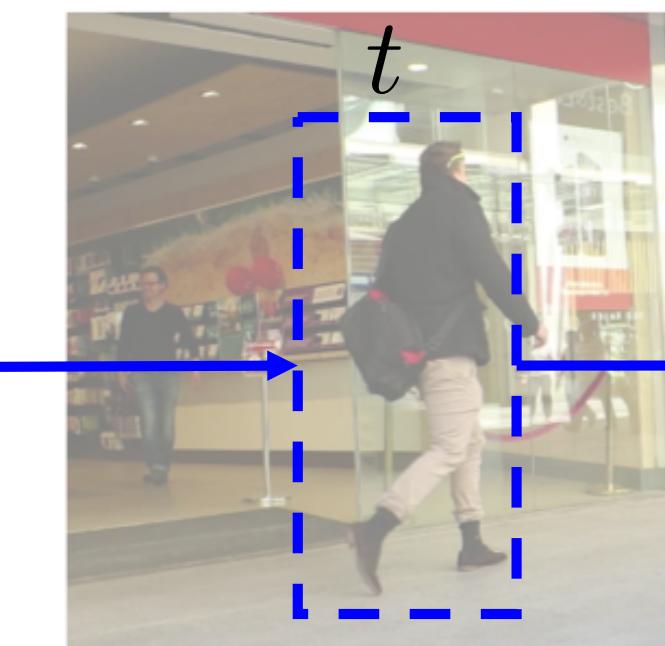
1. Detect objects in frame $k - 1$ (e.g. using an object detector)
2. Initialise in the same location in frame k
3. **Refine** predictions with regression
4. Run object detection again to find new tracks

Approach 2: Tracktor

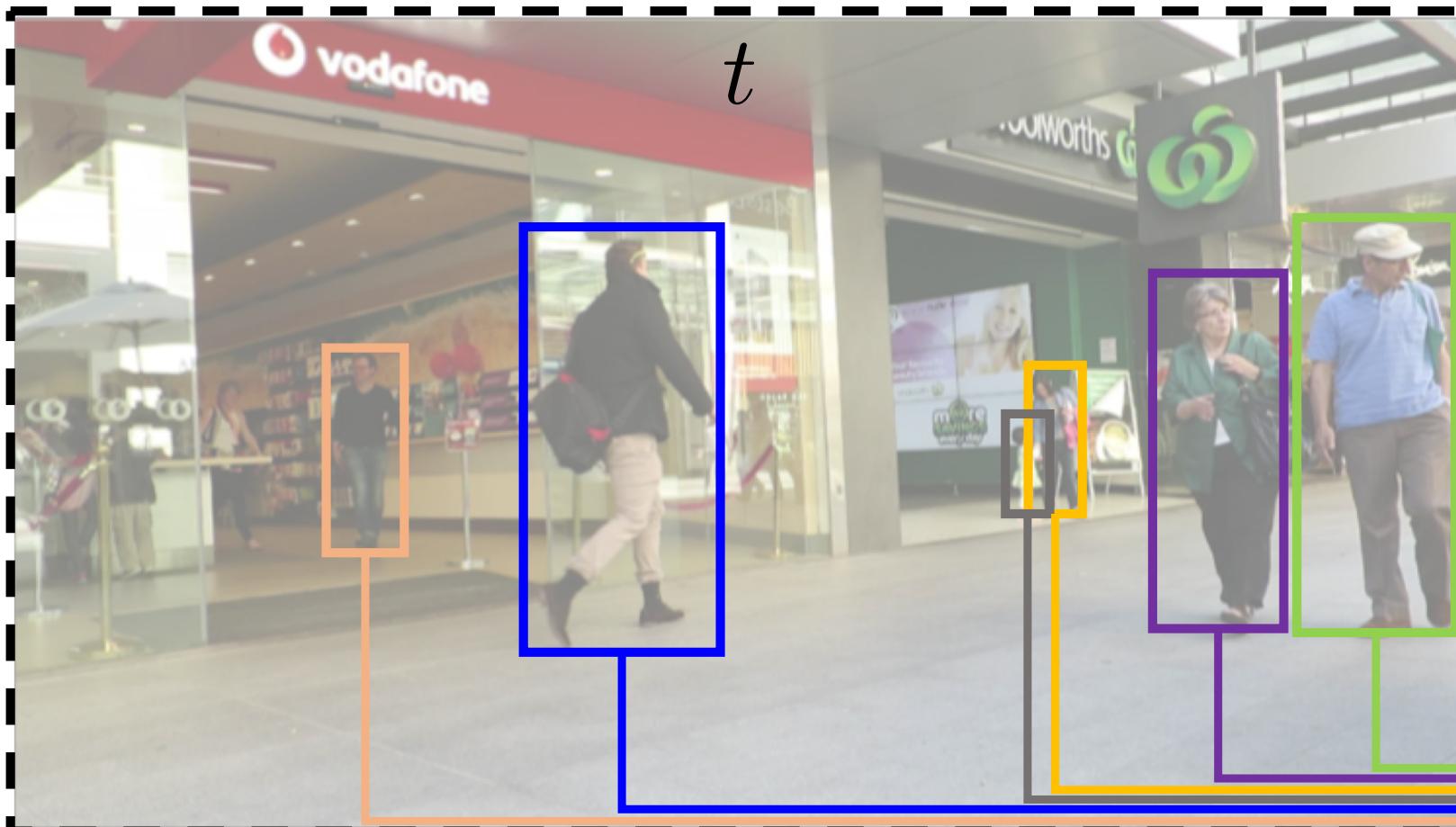
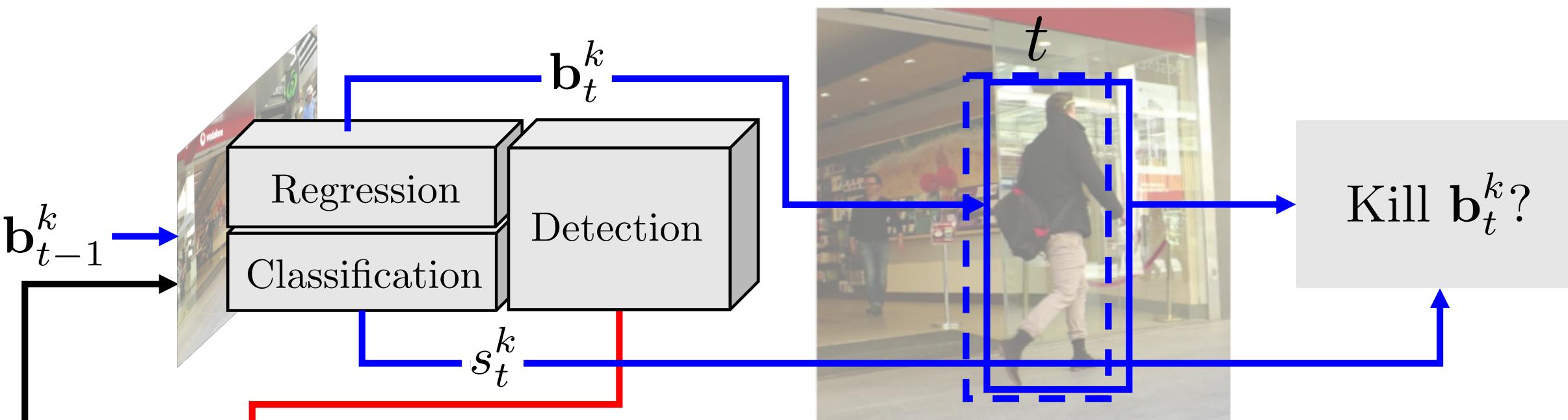
1. Run detection



2. Copy boxes the next frame



3. Refine boxes with regression



4. Initialise new tracks



Bergmann et al., "Tracking without bells and whistles". ICCV 2019

Tracktor: advantages

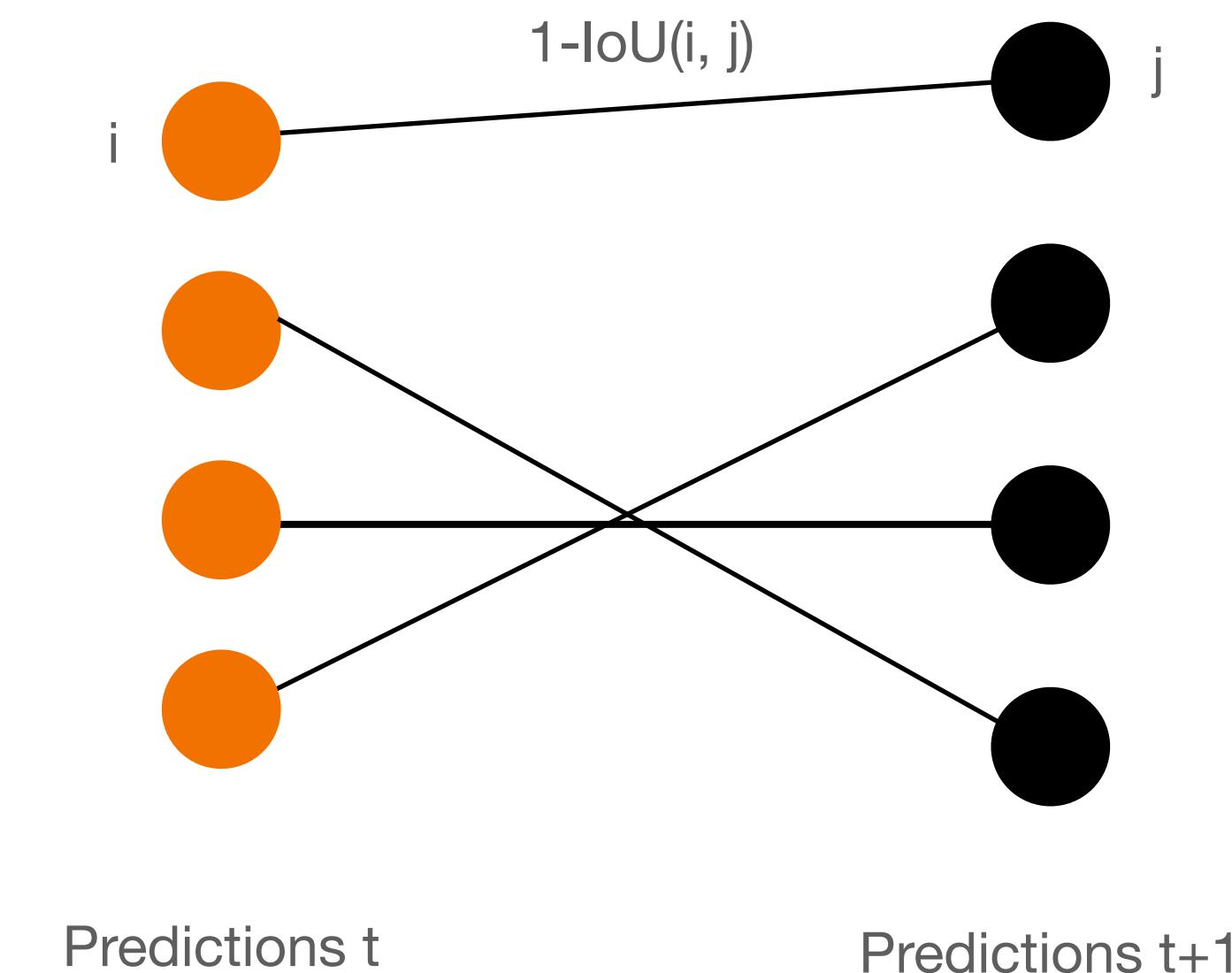
- We can reuse well-engineered object detectors
 - the same architecture of regression and classification heads
- Works well even if trained on still images! (Quiz: Why?)
 - The regression head is agnostic to the object ID and category.

Tracktor: disadvantages

- No motion model
 - problems due to large motions (camera, objects) or low frame rate
- Confusion in crowded spaces
 - since there is no notion of “identity” in the model
- Temporary occlusions (the track is “killed”)
 - generally applies to all online trackers
 - partial remedy: long-term memory of tracks (using object embeddings)

Open questions

- How to match motion prediction with detections?
 - in general: reduction to the assignment problem;
 - small motion: refine the boxes with a regression head
- We have used IoU so far
 - implicit assumption of small motion
- We need a more robust **metric!**



Metric learning for re-identification (Re-ID)

Metric spaces

Definition. A set X (e.g. containing images) is said to be a **metric space** if with any two points p and q of X there is associated a real number $d(p, q)$, called the **distance** from p to q , such that

- $d(p, q) > 0$ if $p \neq q$; $d(p, p) = 0$;
- $d(p, q) = d(q, p)$
- $d(p, q) \leq d(p, r) + d(r, q)$ for any $r \in X$.

Any function with these properties is called a distance function, or a metric.

Walter Rudin, “Principles of mathematical analysis”.

How do we learn a metric space?

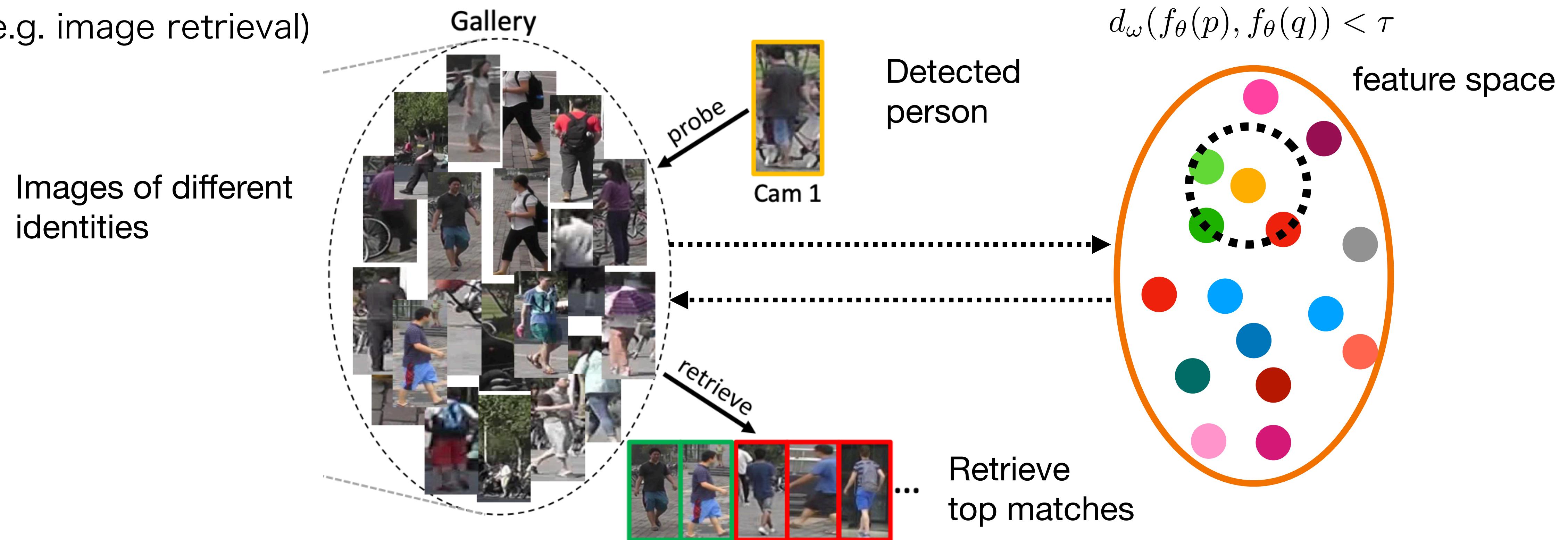
Let us reformulate:

$$d(p, q) = d_\omega(f_\theta(p), f_\theta(q))$$

- We can decouple representation from the distance function:
 - we can use simple metrics (e.g. L2, L1, etc.) or parametric (Mahalanobis distance);
- The problem reduces to learning a feature representation $f_\theta(\cdot)$.

Related applications

- Many problems can be reduced to metric learning
- (e.g. image retrieval)

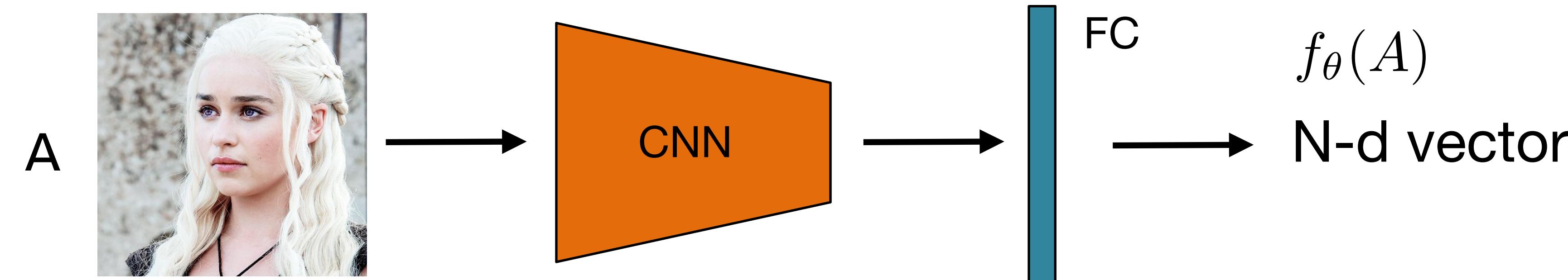


Metric learning

- How do we train a network to learn a feature representation?

Metric learning

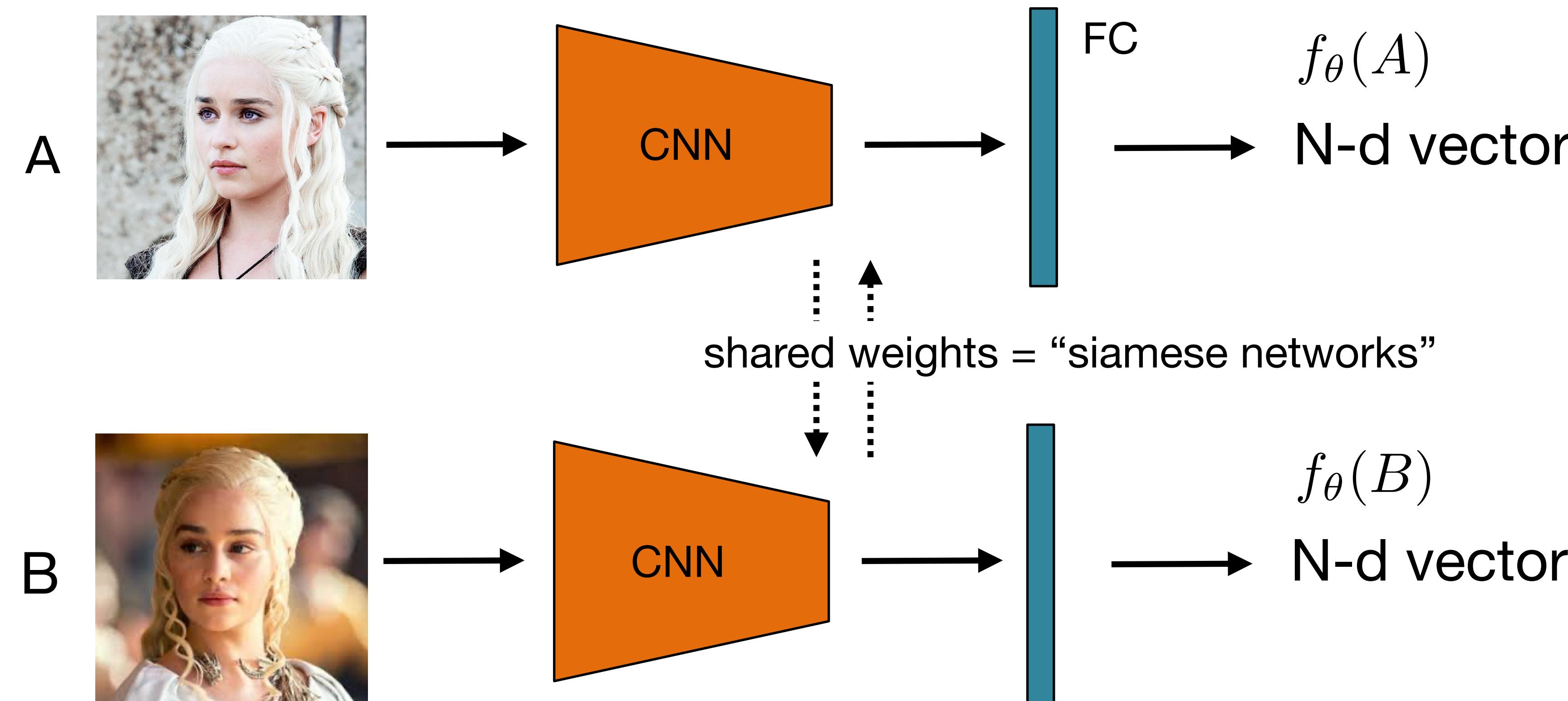
- How do we train a network to learn a feature representation?



Taigman et al. „DeepFace: closing the gap to human level performance“. CVPR 2014

Similarity Learning

- How do we train a network to learn a feature representation?



Chopra et al., “Learning a Similarity Metric Discriminatively, with Application to Face Verification” (CVPR 2005).

Metric learning

- Choose a distance function, e.g., L2:

$$d(A, B; \theta) := \|f_\theta(A) - f_\theta(B)\|^2$$

- Minimise the distance between image pairs of the same person:

$$\theta^* := \arg \min_{\theta} \mathbb{E}_{A,B}[d(A, B; \theta)]$$

- Quiz: Simple, but are we done?

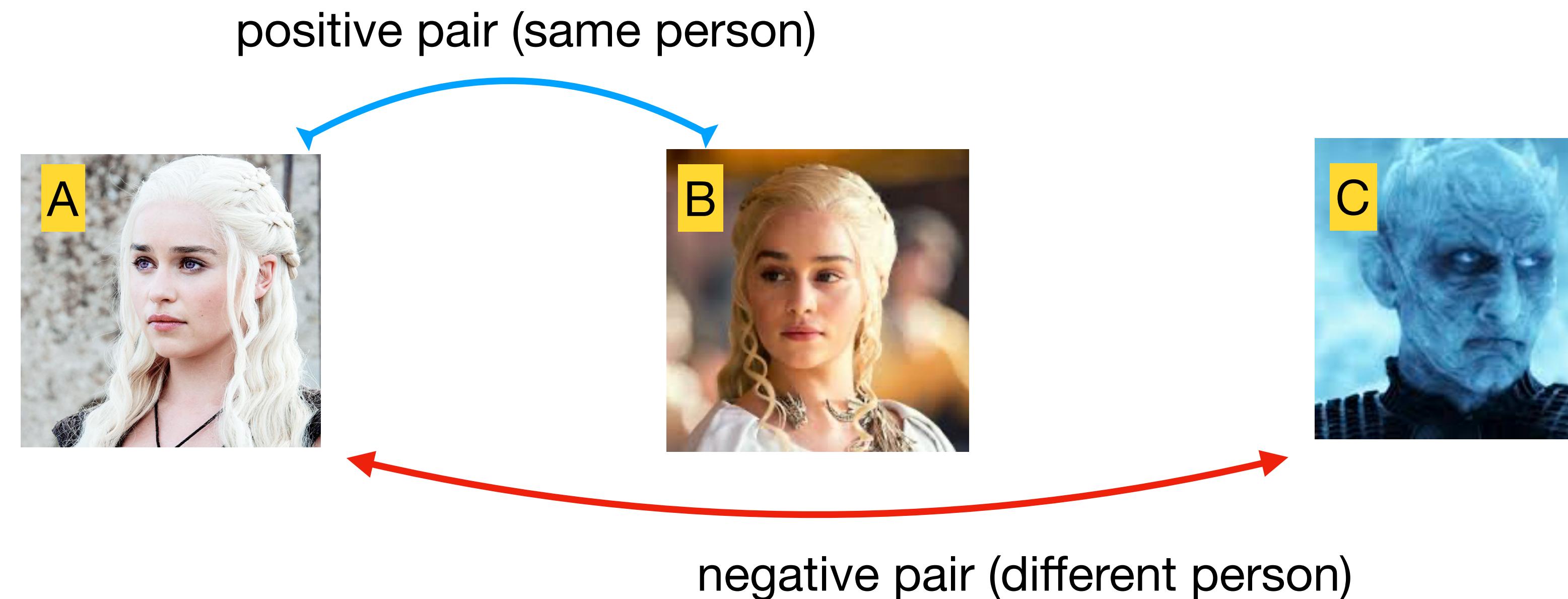
Metric learning

- We can simply “learn” $d(A, B; \theta) = 0$ for all images!
- How can we prevent this?



- We can add negative pairs.
 - minimise the distance between positive pairs; maximise it otherwise.

Metric learning: loss functions



Our goal: $d(A, B; \theta) < d(A, C; \theta)$

Metric learning: loss functions

Our goal: $d(A, B; \theta) < d(A, C; \theta)$

The loss (second attempt):

$$\theta^* := \arg \min_{\theta} \mathbb{E}_{A, B \in S^+} [d_\theta(A, B)] - \mathbb{E}_{B, C \in S^-} [d_\theta(B, C)]$$

S^+ and S^- are a set of positive and negative image pairs.

In practice, the loss is a little more sophisticated...

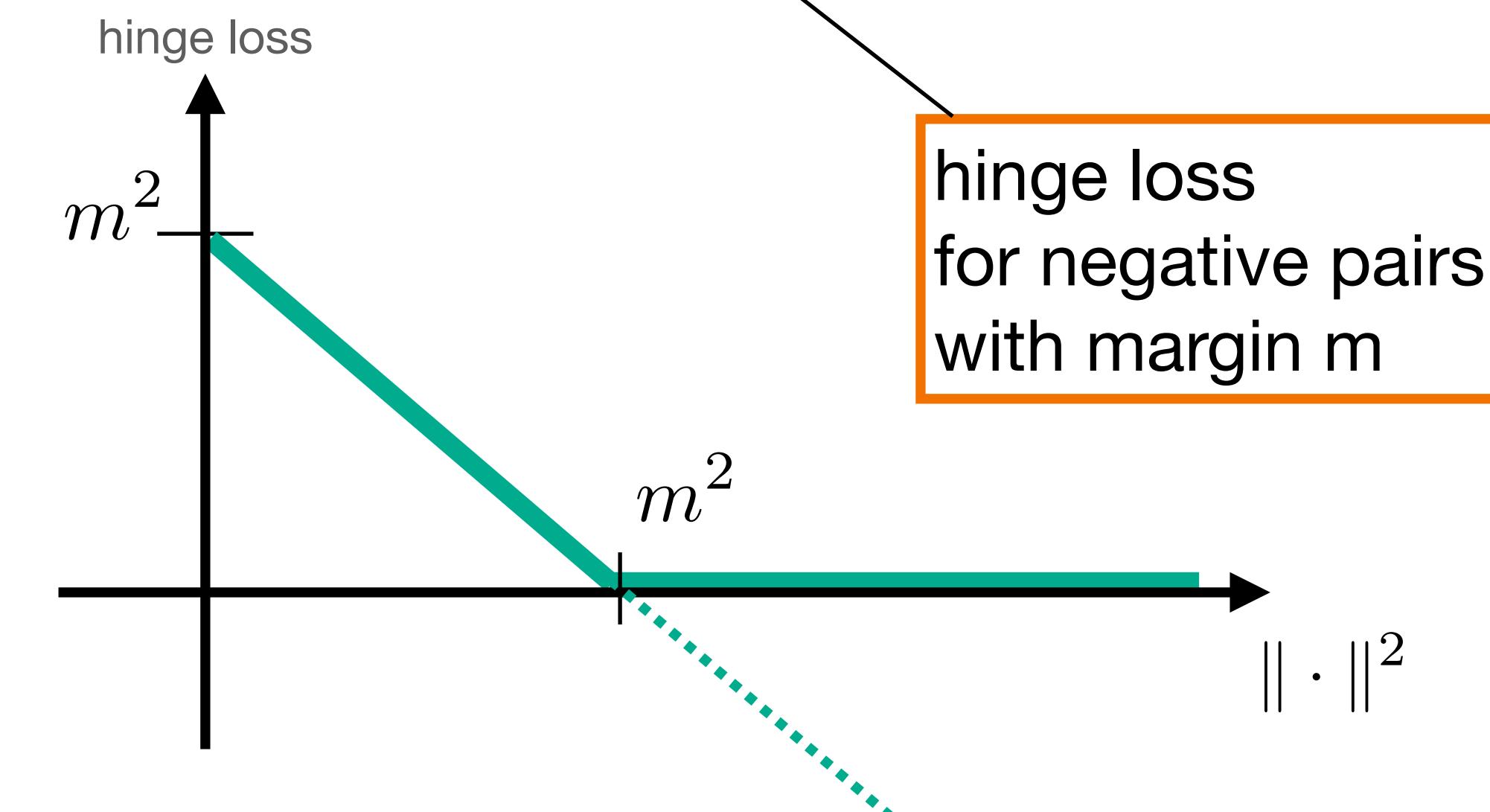
Metric learning: hinge loss

Unbounded loss for positive pairs, bounded for negative pairs:

$$\mathcal{L}(A, B) = y^* \|f(A) - f(B)\|^2 + (1 - y^*) \max(0, m^2 - \|f(A) - f(B)\|^2)$$

1 if (A,B) is a positive pair
0 otherwise

L2 distance



hinge loss
for negative pairs
with margin m

Chopra et al., "Learning a Similarity Metric Discriminatively, with Application to Face Verification" (CVPR 2005).

Metric learning: triplet loss

Idea: implement the inequality $d(A, B; \theta) < d(A, C; \theta)$ in a single loss term:



Anchor



Positive



Negative

Triplet loss:

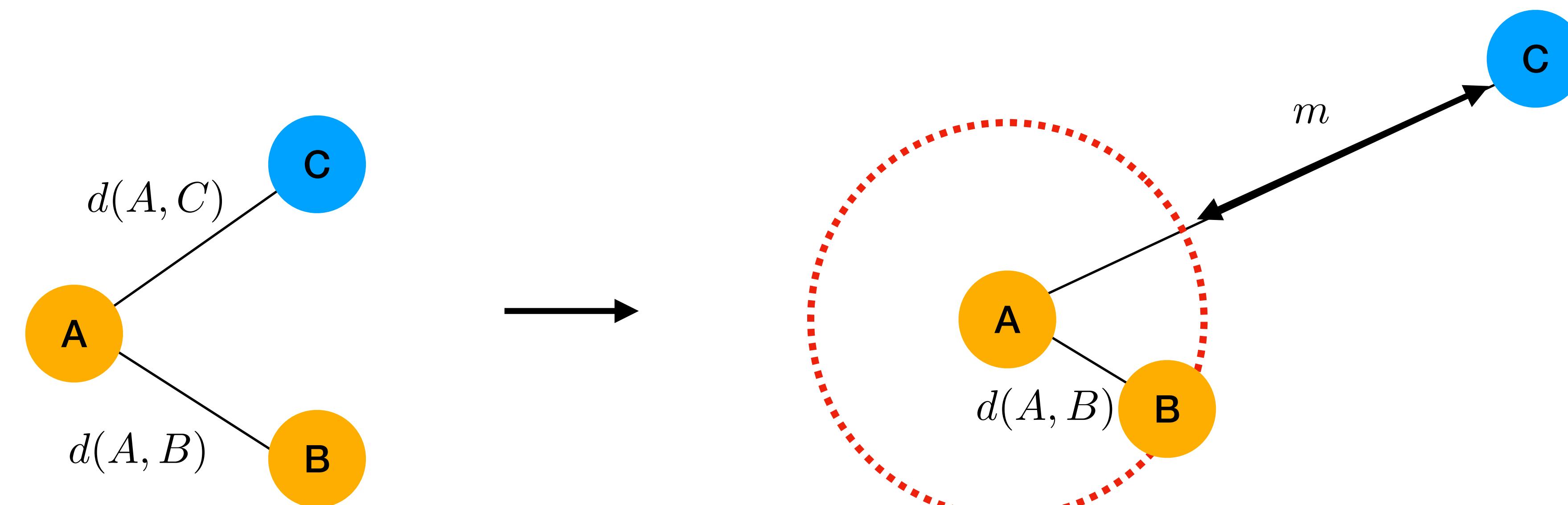
$$\mathcal{L}(A, B, C) = \max(0, \|f(A) - f(B)\|^2 - \|f(A) - f(C)\|^2 + m)$$

margin

Metric learning: triplet loss

$$\mathcal{L}(A, B, C) = \max(0, \|f(A) - f(B)\|^2 - \|f(A) - f(C)\|^2 + m)$$

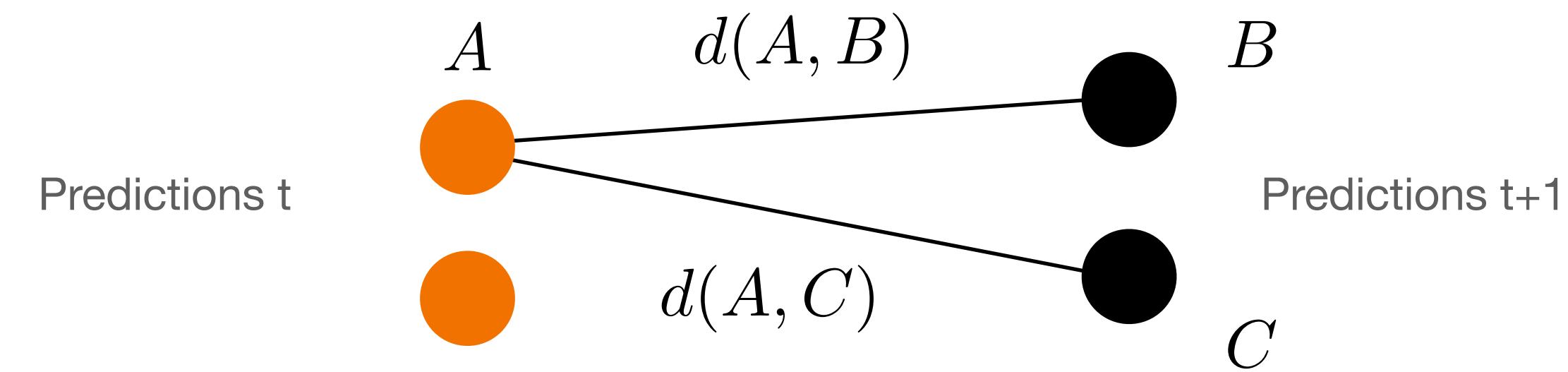
Intuitive idea:



Metric learning for tracking

How does this help tracking?

1. Train an embedding network on triplets of data:
 - positive pairs: same person at different timesteps;
 - negative pairs: different people;
2. Use the network to compute the similarity score for matching



Metric learning: summary

- Many problems can be reduced to metric learning,
 - including MOT (both online and offline).
- Annotation needed:
 - e.g. same identity in different images.
- In practice: careful tuning of the positive pair term vs. the negative term;
 - hard-negative mining and a bounded function for the negative pairs help;
- Extension to unsupervised setting – contrastive learning (later in the course);
- Next lecture: offline tracking.

What we've learned today

- Bayesian tracking (a general framework).
- Deep single object trackers:
 - GOTURN, MDNet (online adaptation).
- Tracking by detection:
 - Data association problem; bipartite matching.
- Metric learning
 - Hinge and triplet loss

Additional reading (optional)

- Ning et al., „Spatially Supervised Recurrent Convolutional Neural Networks for Visual Object Tracking“ (ISCAS 2017).
- Chopra et al., “Learning a Similarity Metric Discriminatively, with Application to Face Verification” (CVPR 2005).
- Elezi et al., “The Group Loss for Deep Metric Learning” (ECCV 2020).
- Manmatha et al., “Sampling matters for deep metric learning” (ICCV 2017).
- Xu et al., “Deep asymmetric metric learning via rich relationship mining” (CVPR 2019).
- Duan et al., “Deep embedding learning with discriminative sampling policy” (CVPR 2019).
- Wang et al., “Ranked list loss for deep metric learning” (CVPR 2019).
- Wang et al., “Multi-similarity loss with general pair weighting for deep metric learning” (CVPR 2019)