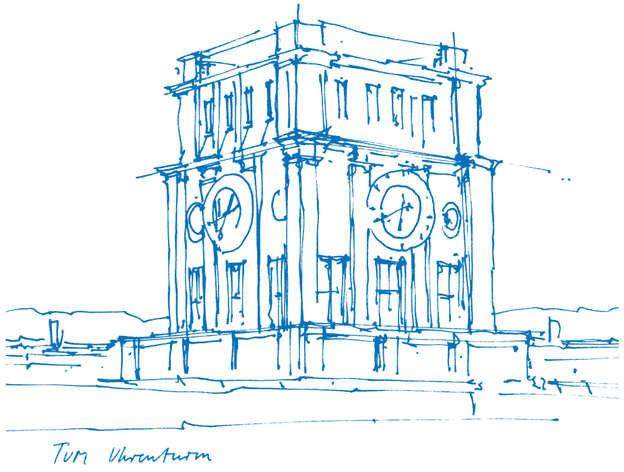
Computer Vision III:

MOT 02: Offline tracking

Dr. Nikita Araslanov
21.11.2023

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



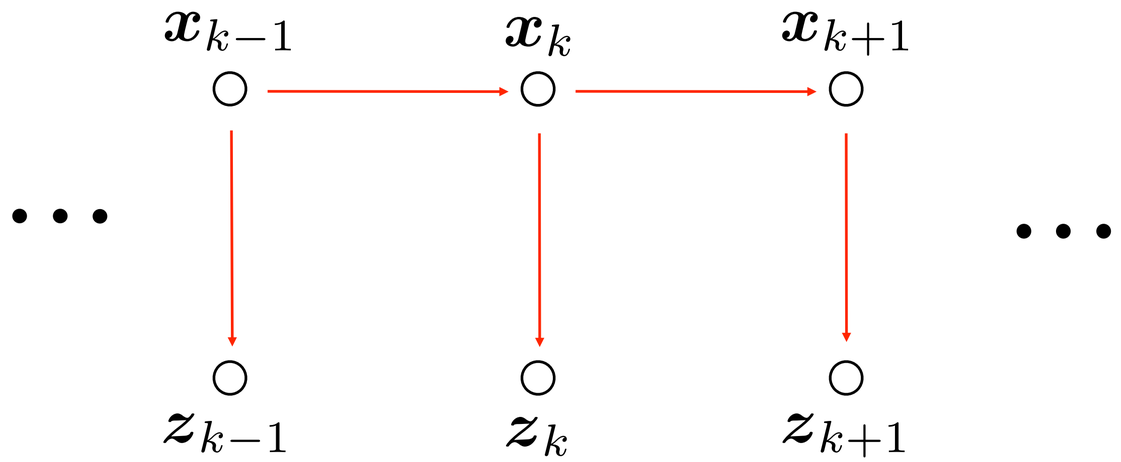
Problem statement

- Our focus: object detection and temporal association

frame t frame $t + n$

Bayesian graphical model

- Hidden Markov model:



Assumptions:

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

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

[Stefan Roth]

Bayesian formulation

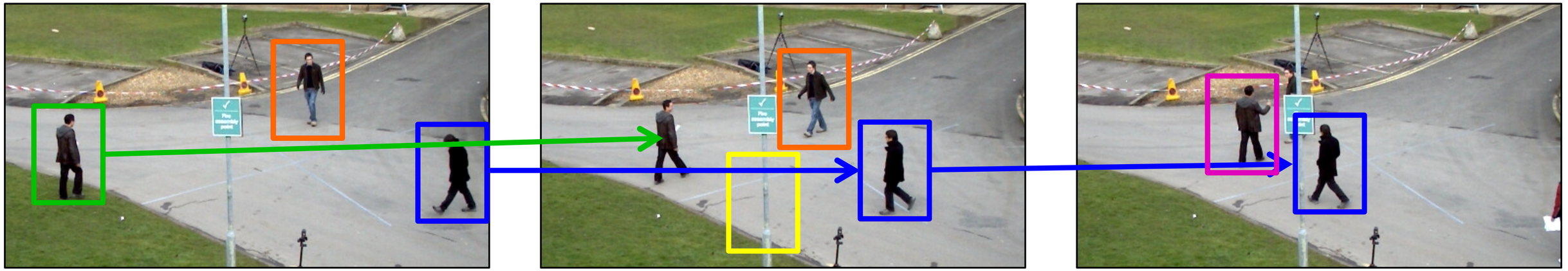
$$p(\boldsymbol{x}_k|\mathbf{Z}_k) = \kappa \cdot p(\boldsymbol{z}_k|\boldsymbol{x}_k) \cdot \int p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) \cdot p(\boldsymbol{x}_{k-1}|\mathbf{Z}_{k-1}) \, \mathrm{d}\boldsymbol{x}_{k-1}$$

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

[Stefan Roth]

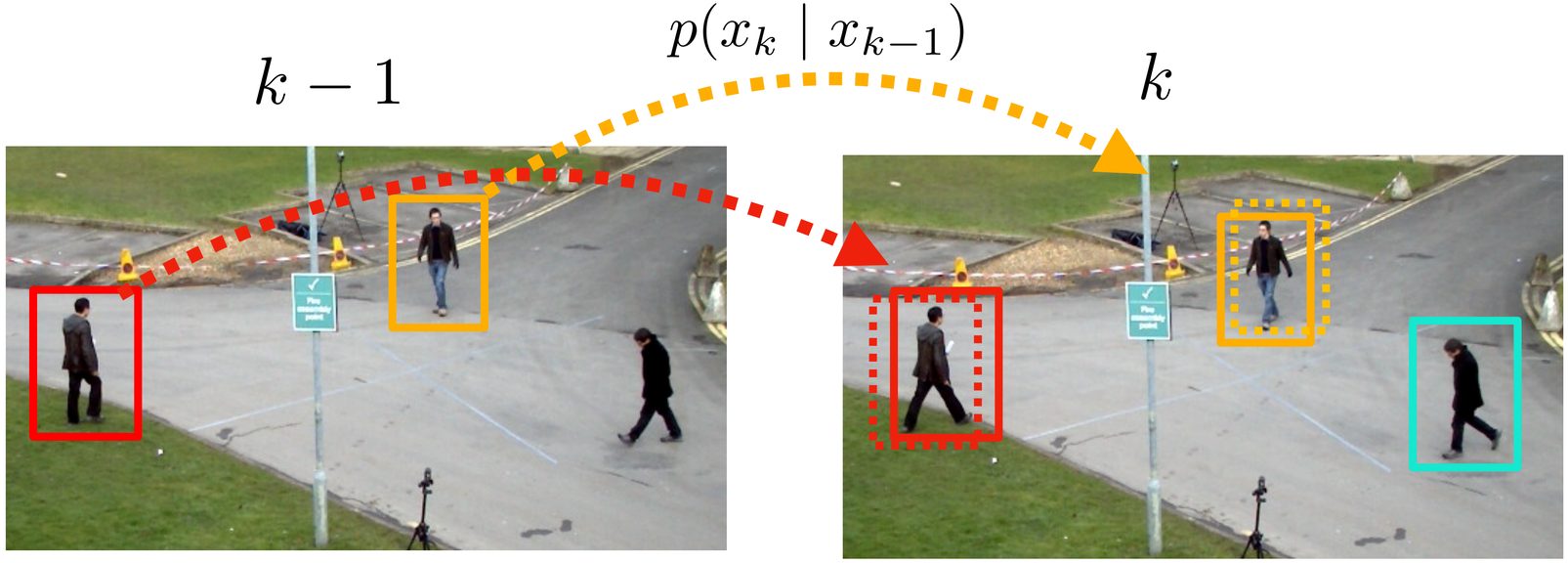
Tracking-by-detection

- We will focus on algorithms where a set of detections is provided.
- Remember detections are not perfect!



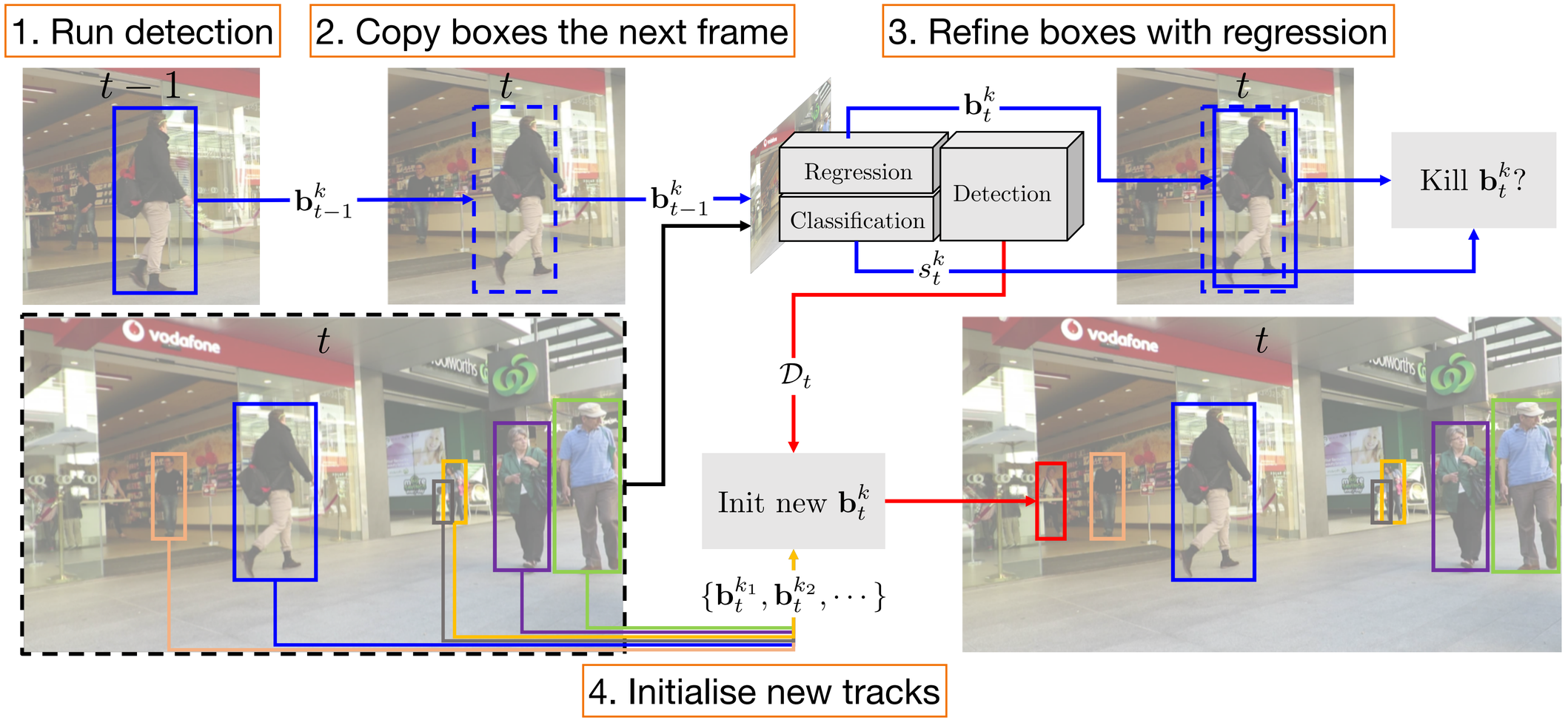
Find detections that match and form a trajectory

Approach 2 – “copy, refine, detect new”



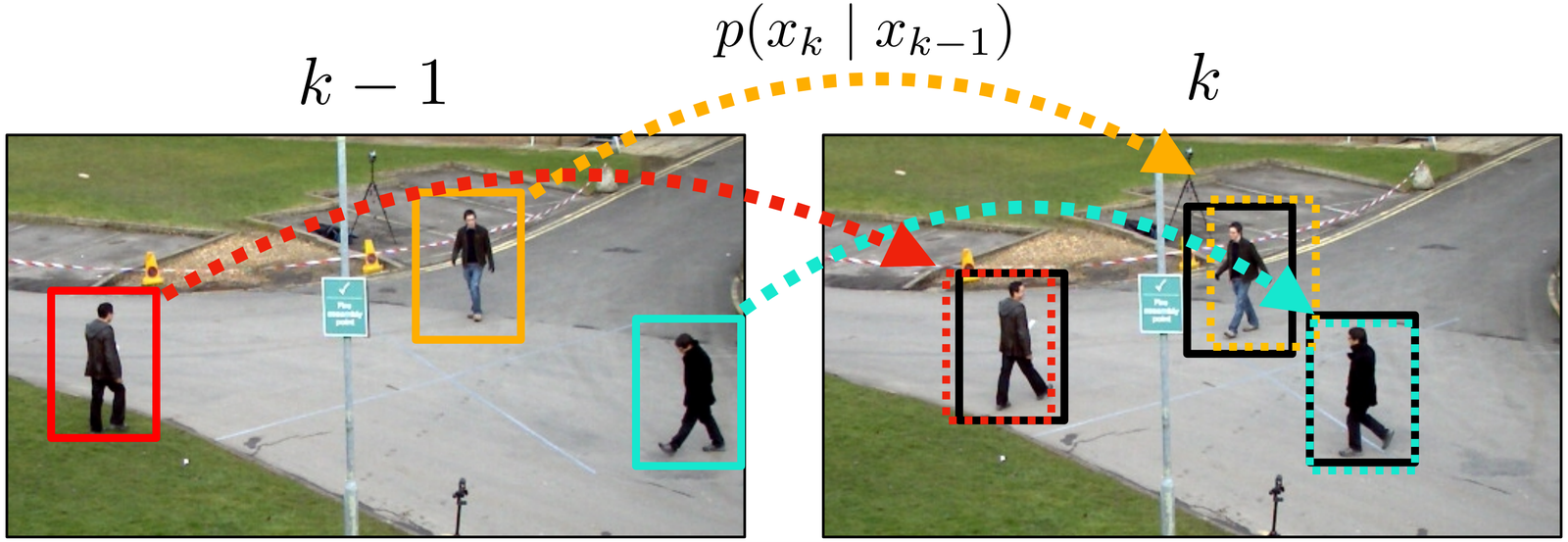
1. Detect objects in frame $k-1$ (e.g. using an object detector)
2. Initialise in the same location in frame k
3. Refine predictions with regression
4. Run object detection again to find new tracks

Approach 2: Tracktor



Bergmann at al., “Tracking without bells and whistles”. ICCV 2019

Approach 1 – “detect all and match”



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

Bipartite matching

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

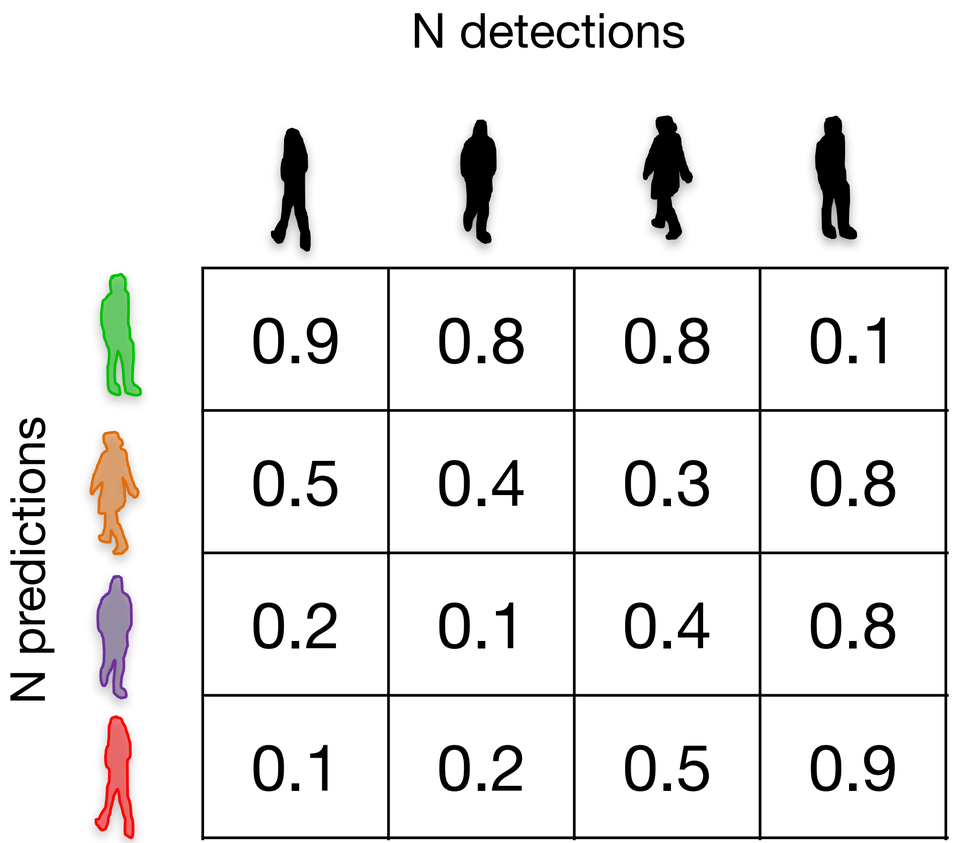
N detections

N predictions

0.9	0.8	0.8	0.1
0.5	0.4	0.3	0.8
0.2	0.1	0.4	0.8
0.1	0.2	0.5	0.9

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

N detections

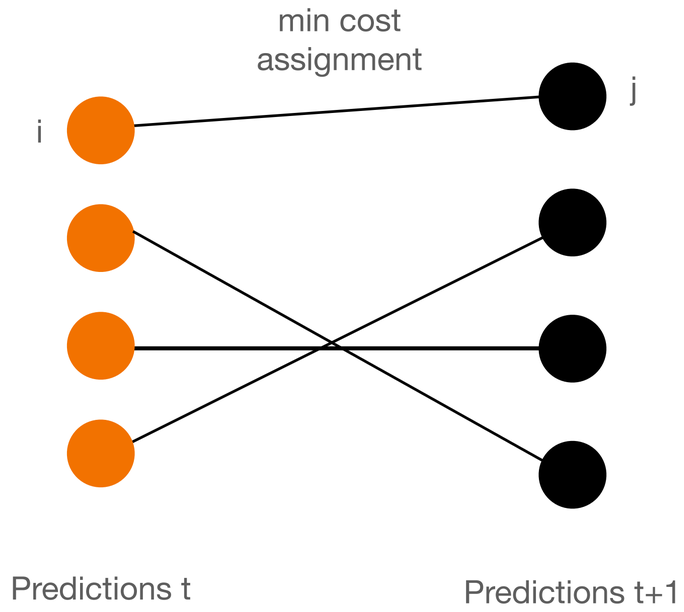
N predictions

0.9	0.8	0.8	**0.1**
0.5	0.4	**0.3**	0.8
0.2	**0.1**	0.4	0.8
0.1	0.2	0.5	0.9

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

Open questions

- How to match motion prediction with detections?
  - **in general: reduction to the assignment problem;**
- We have used IoU so far
  - implicit assumption of small motion
- We could use 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:

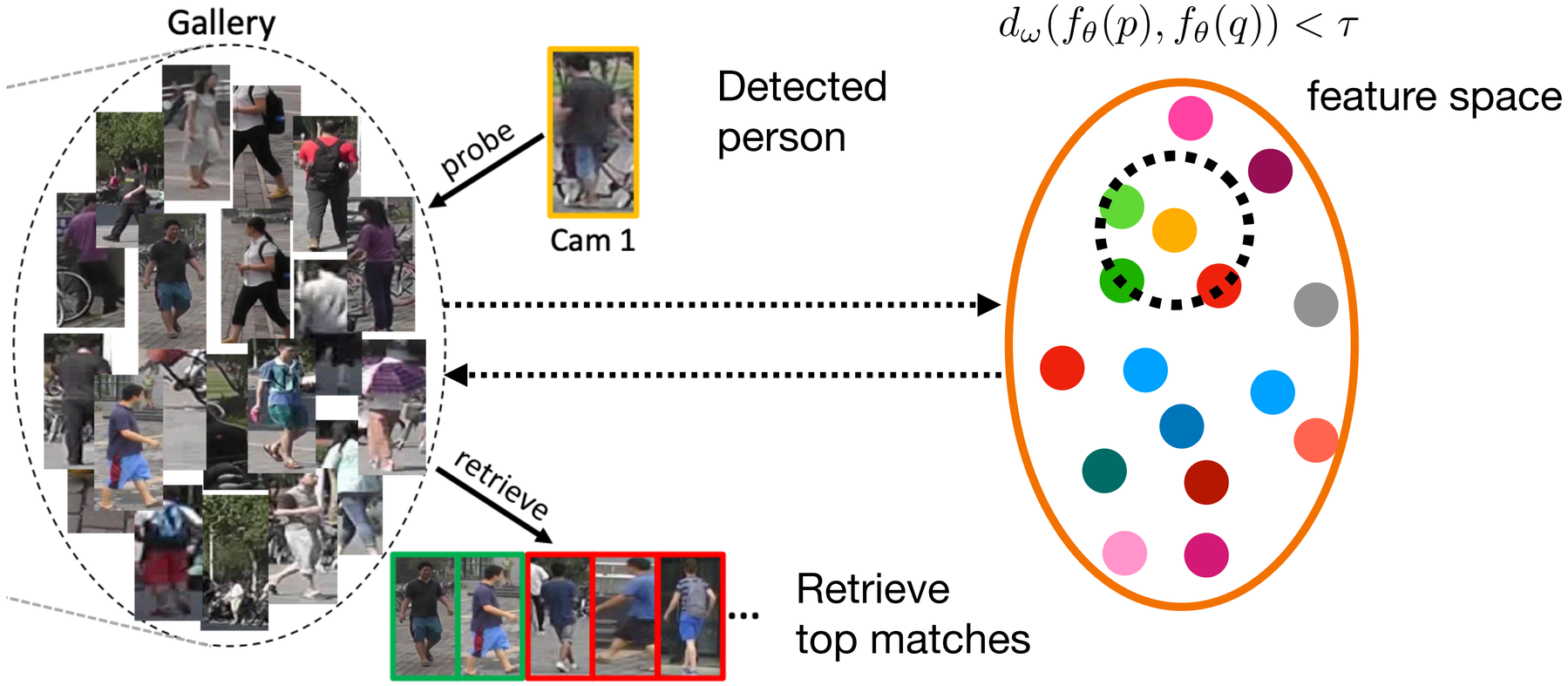
$$d(p, q) = d_\omega(f_\theta(p), f_\theta(q))$$

- 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)

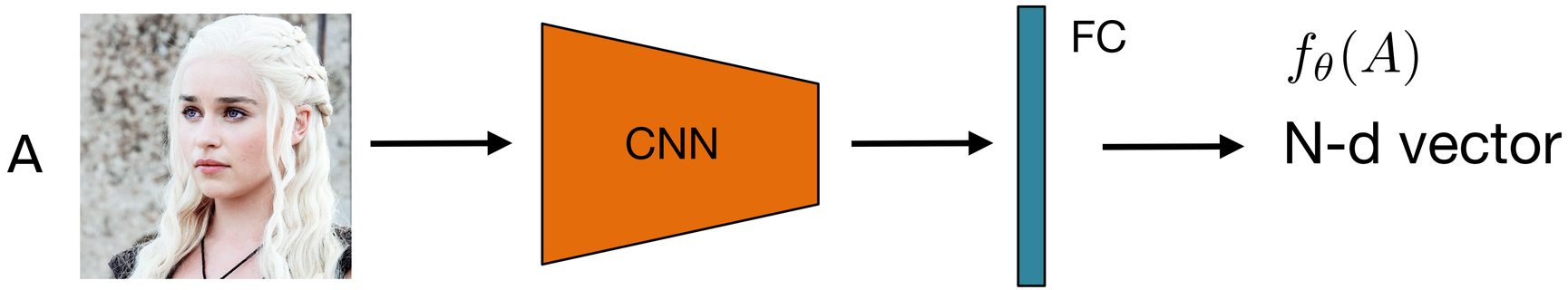


Metric learning

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

Metric learning

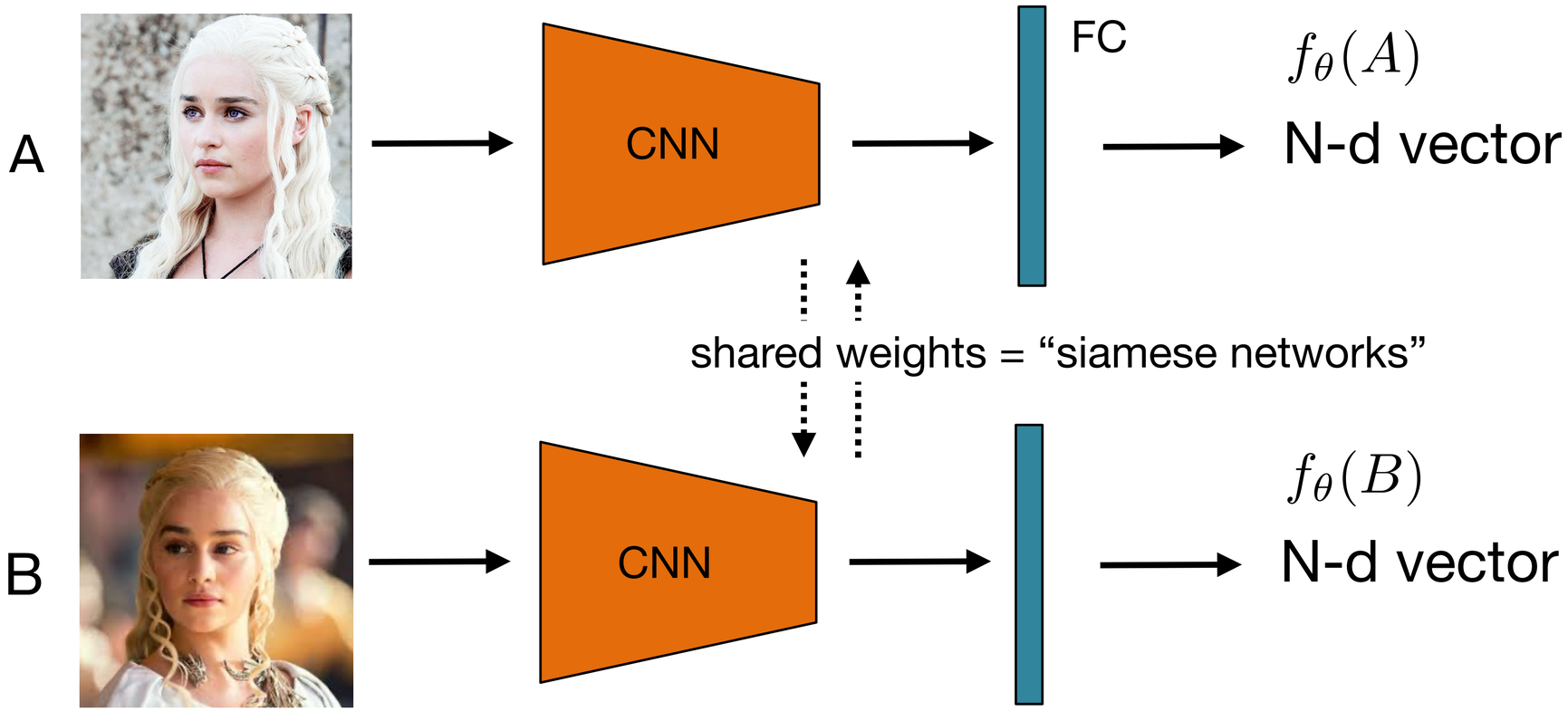
- How do we train a network to learn a feature representation?



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

Metric 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:

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

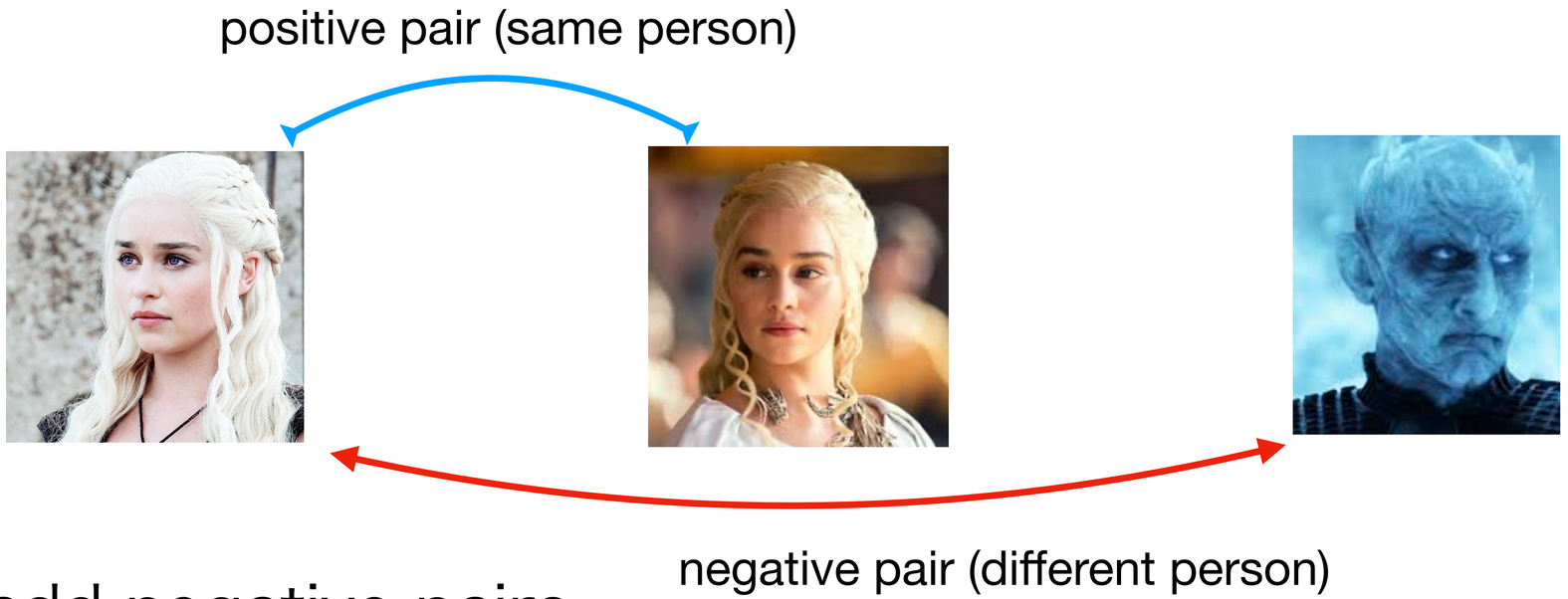
- Minimise the distance between image pairs of the same person:

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

- Quiz: Simple, but are we done?

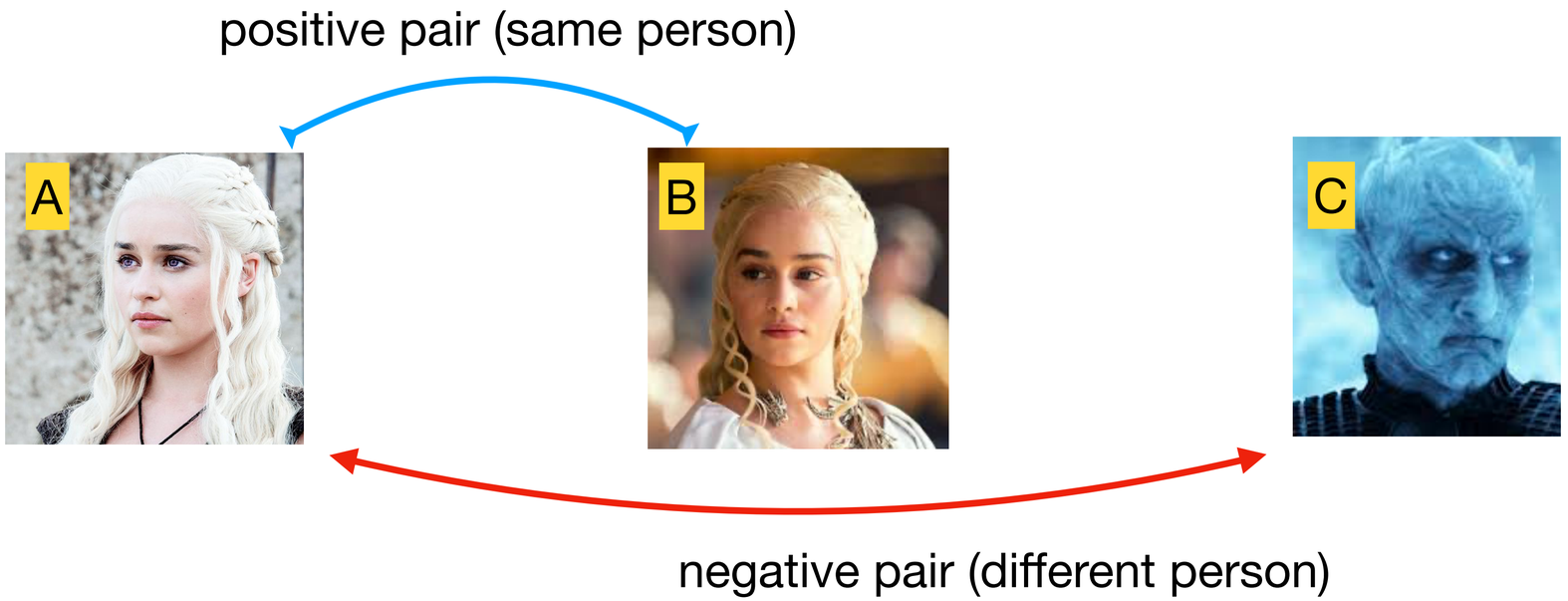
Metric learning

- 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...

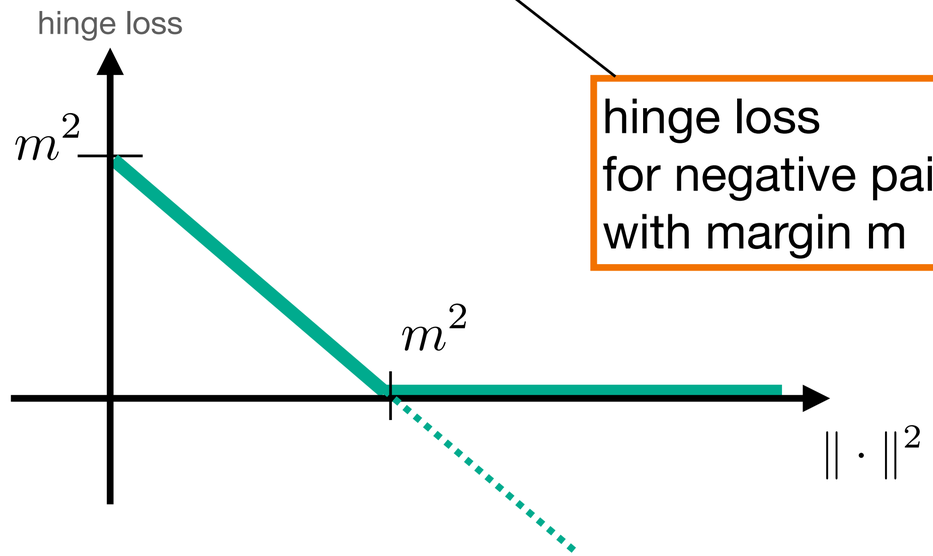
Metric learning: hinge loss

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:

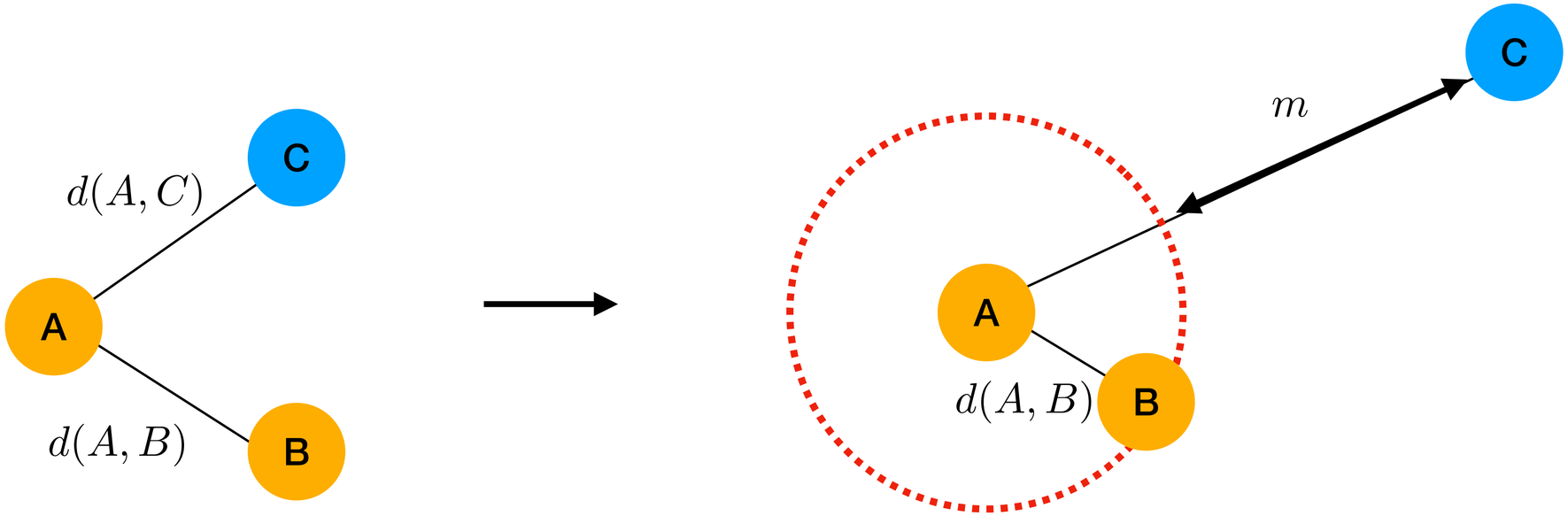
margin

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

Metric learning: triplet loss

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

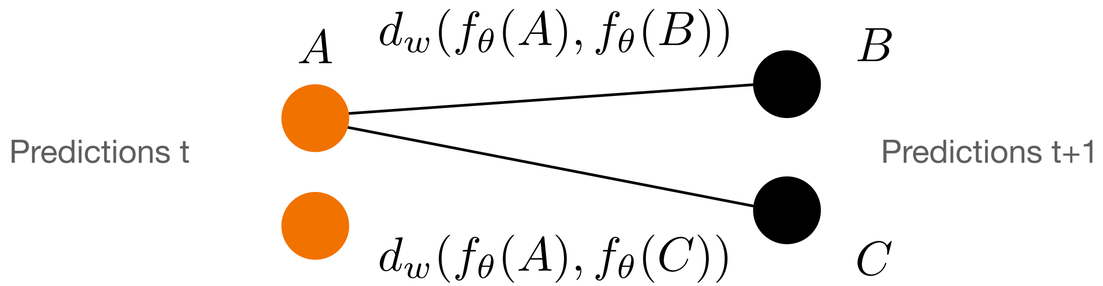
Intuitive idea:



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/cost 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: offline tracking.

Tracking: what we've learned

- 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

Online vs. offline tracking

Last lecture

- Online tracking
  - Processes two frames at a time
  - For real-time applications
  - Prone to drifting → hard to recover from errors or occlusions
- Offline tracking
  - 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

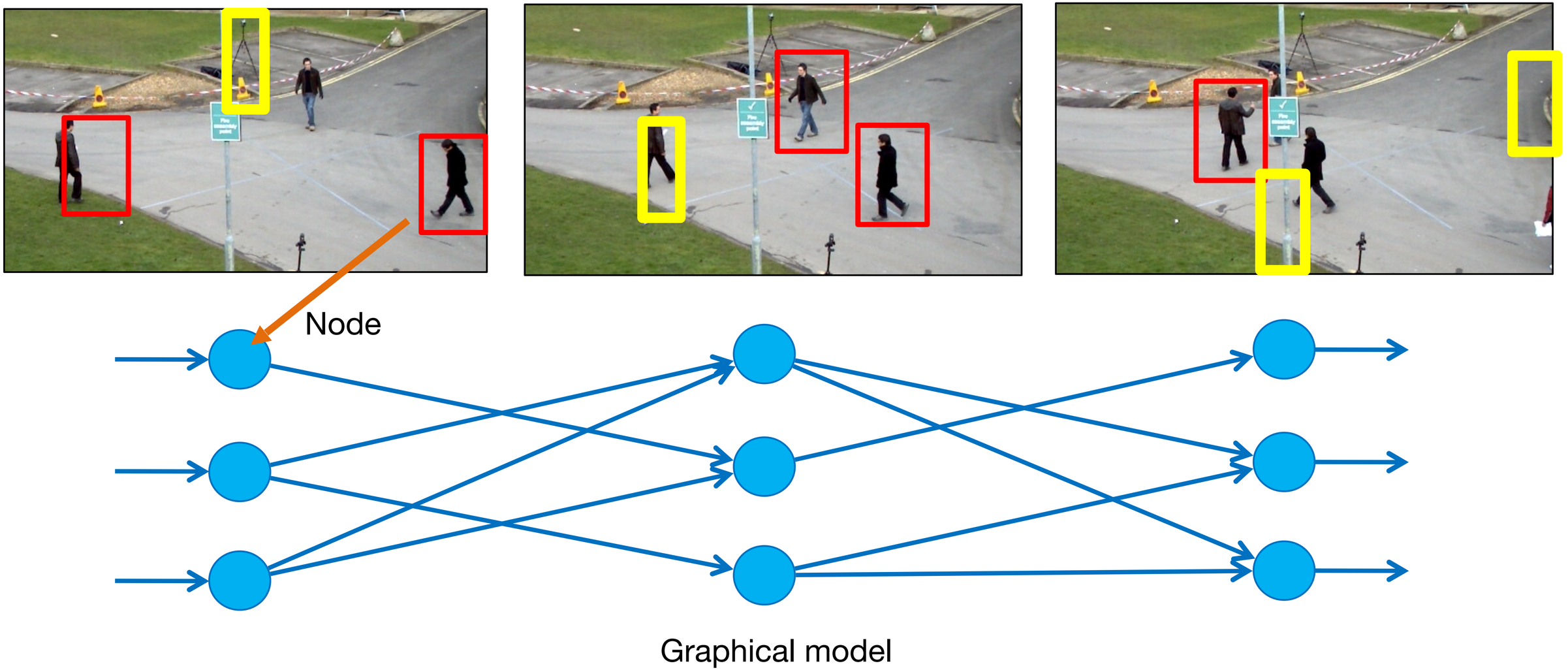
Online vs. offline tracking

- Online tracking
  - Processes two frames at a time
  - For real-time applications
  - Prone to drifting → hard to recover from errors or occlusions
- Offline tracking
  - 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

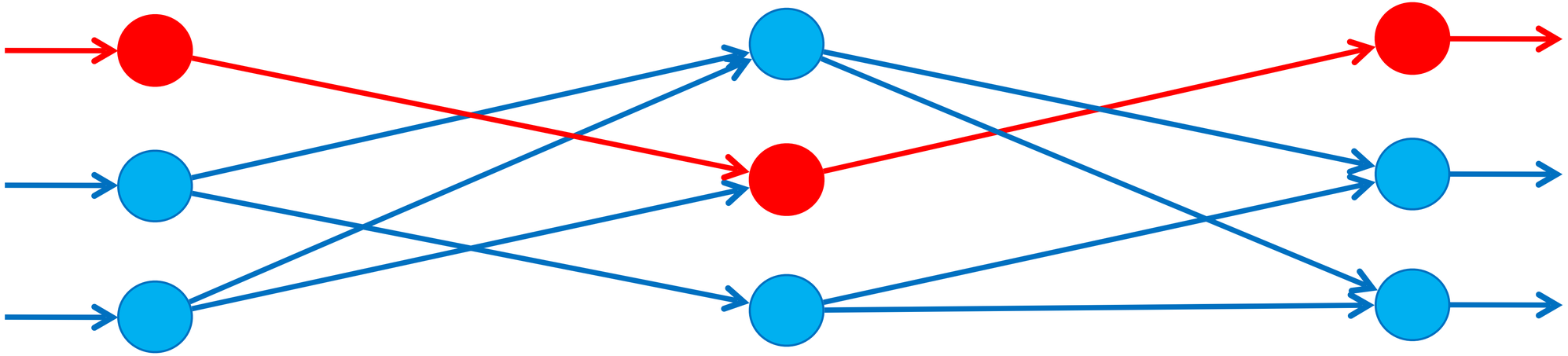
This lecture

Graph-based MOT

Tracking with network flows

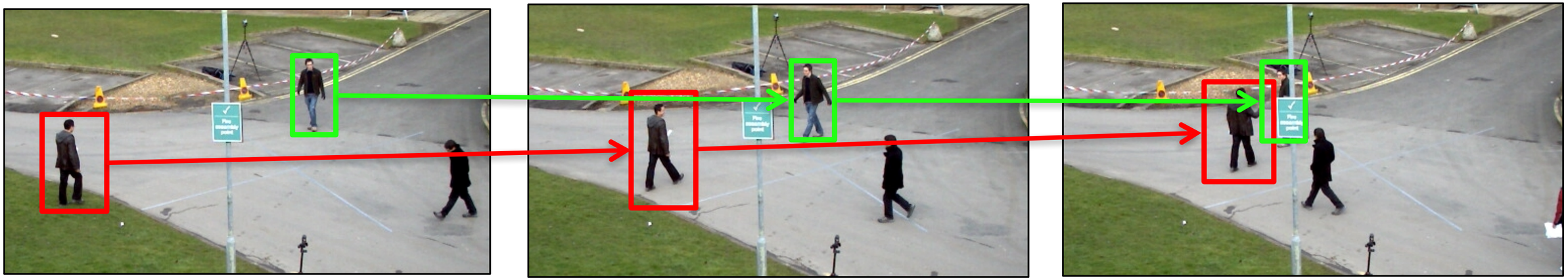


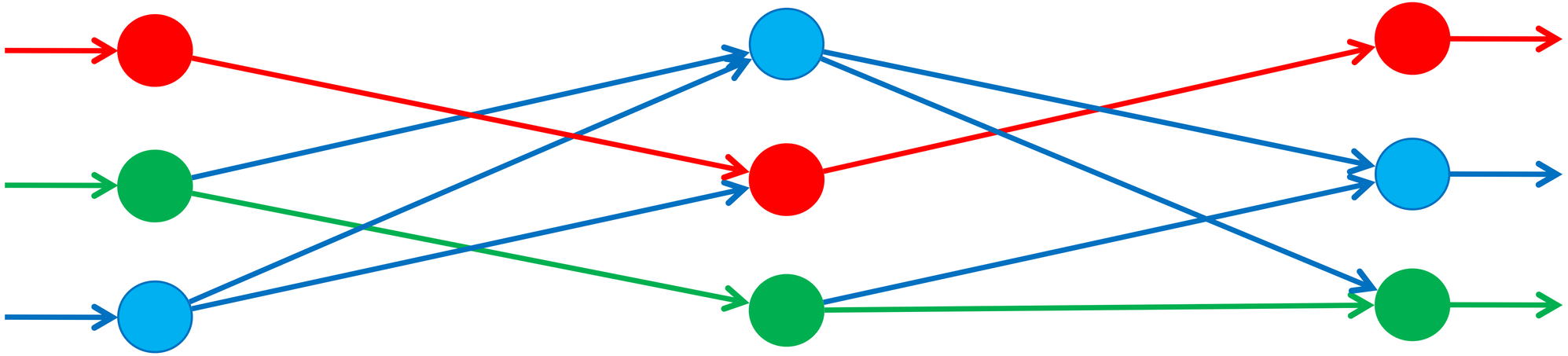
Tracking with network flows



Zhang et al. “Global Data Association for Multi-Object Tracking Using Network Flows“. CVPR 2008

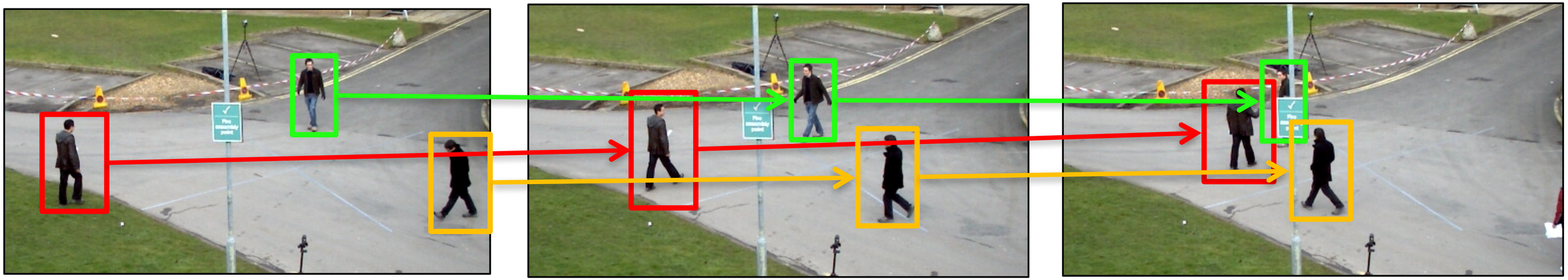
Tracking with network flows

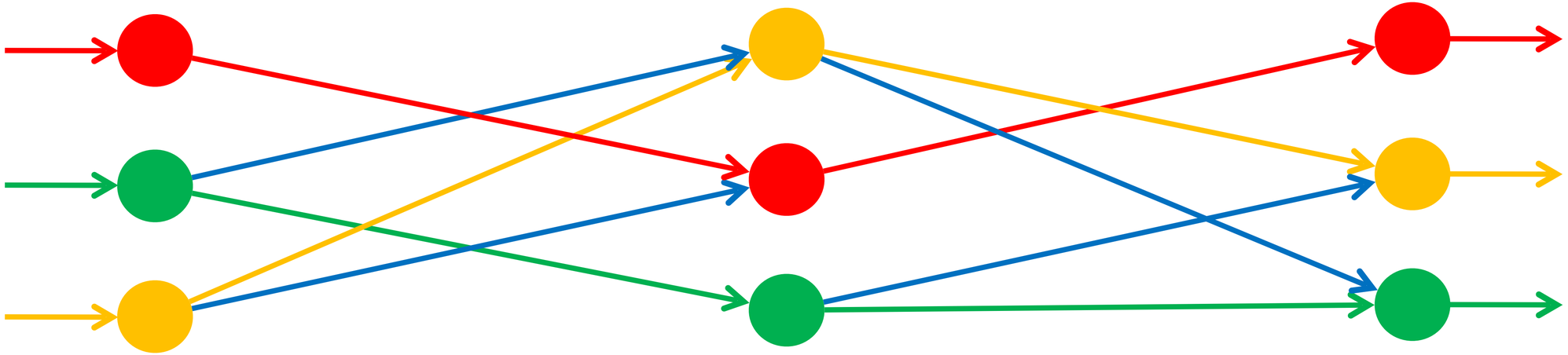




Zhang et al. “Global Data Association for Multi-Object Tracking Using Network Flows“. CVPR 2008

Tracking with network flows





Zhang et al. “Global Data Association for Multi-Object Tracking Using Network Flows“. CVPR 2008

Tracking with network flows

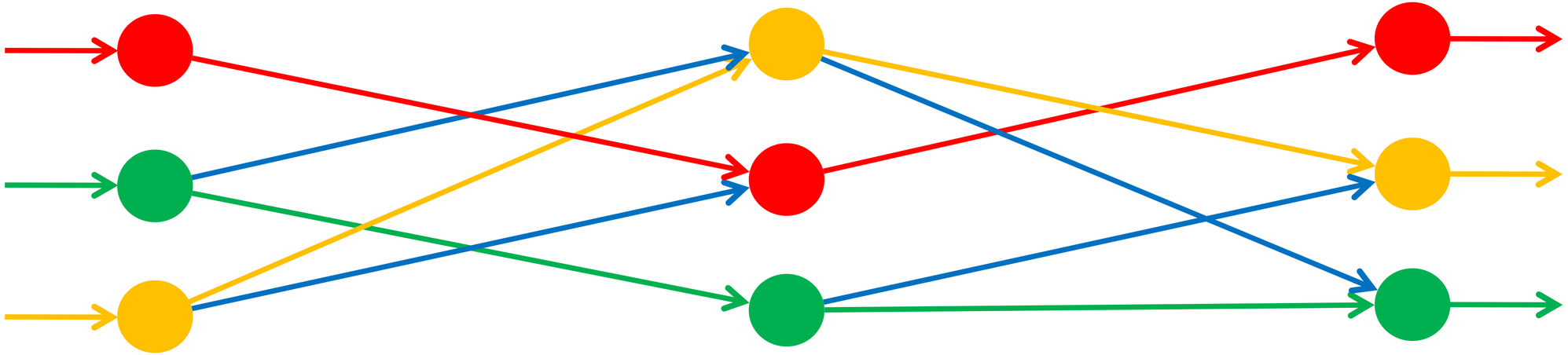
- Node = object detection
- Edge = temporal ID correspondence

- Goal: disjoint set of trajectories

Tracking with network flows

- Minimum-cost maximum-flow problem

"Determine the maximum flow with a minimum cost"



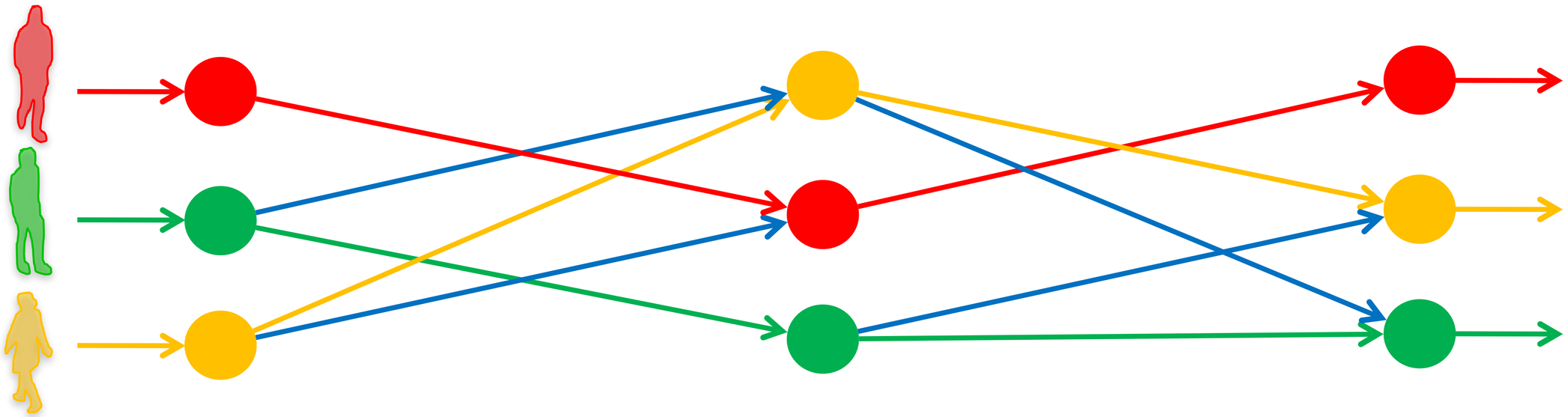
Ravindra K. Ahuja et al., "Network Flows: Theory, Algorithms, and Applications" (1993).

Tracking with network flows

- Minimum-cost maximum-flow problem

"Determine the maximum flow with a minimum cost"

(e.g. 1 unit of flow = 1 pedestrian)



Ravindra K. Ahuja et al., "Network Flows: Theory, Algorithms, and Applications" (1993).

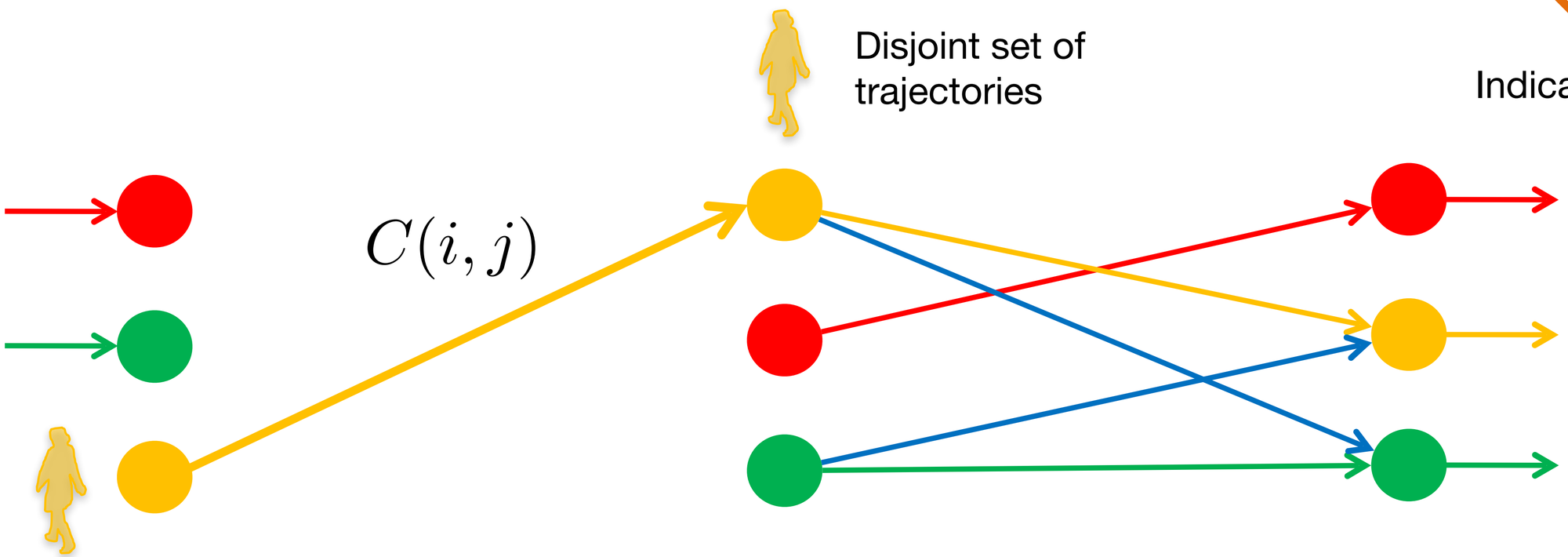
Tracking with network flows

- Minimising the cost:

$$f^* = \arg\min_f \sum_{i,j} C(i,j) f(i,j)$$

Disjoint set of trajectories

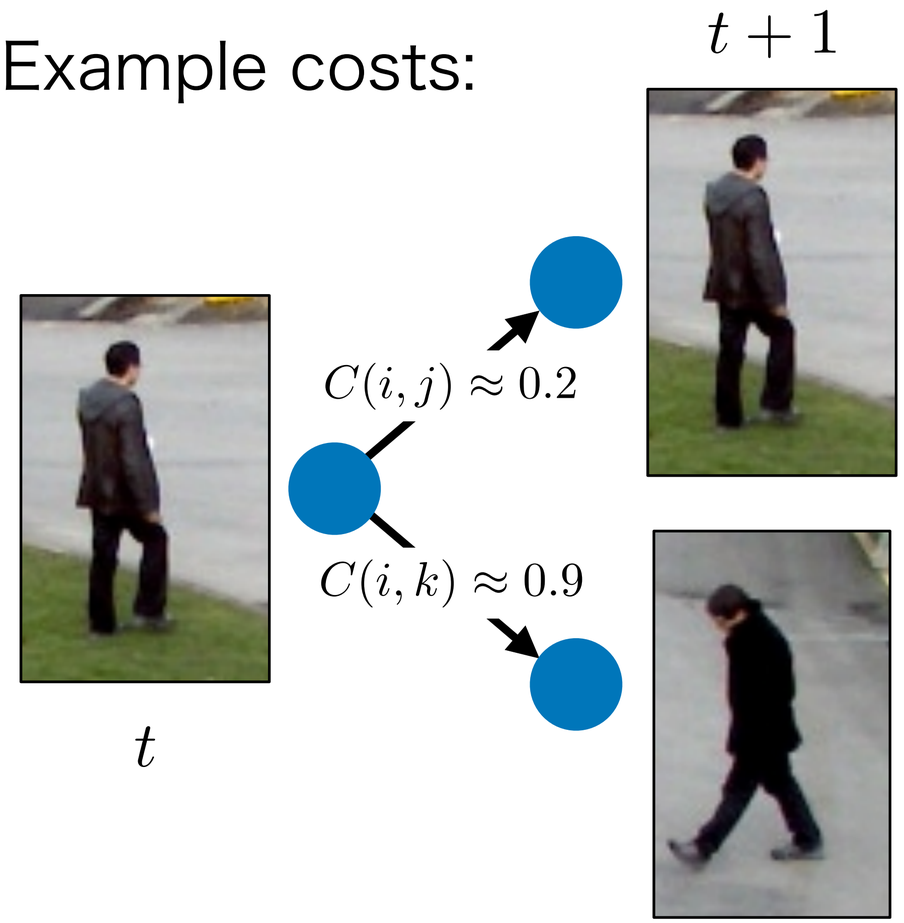
Indicator {0,1}



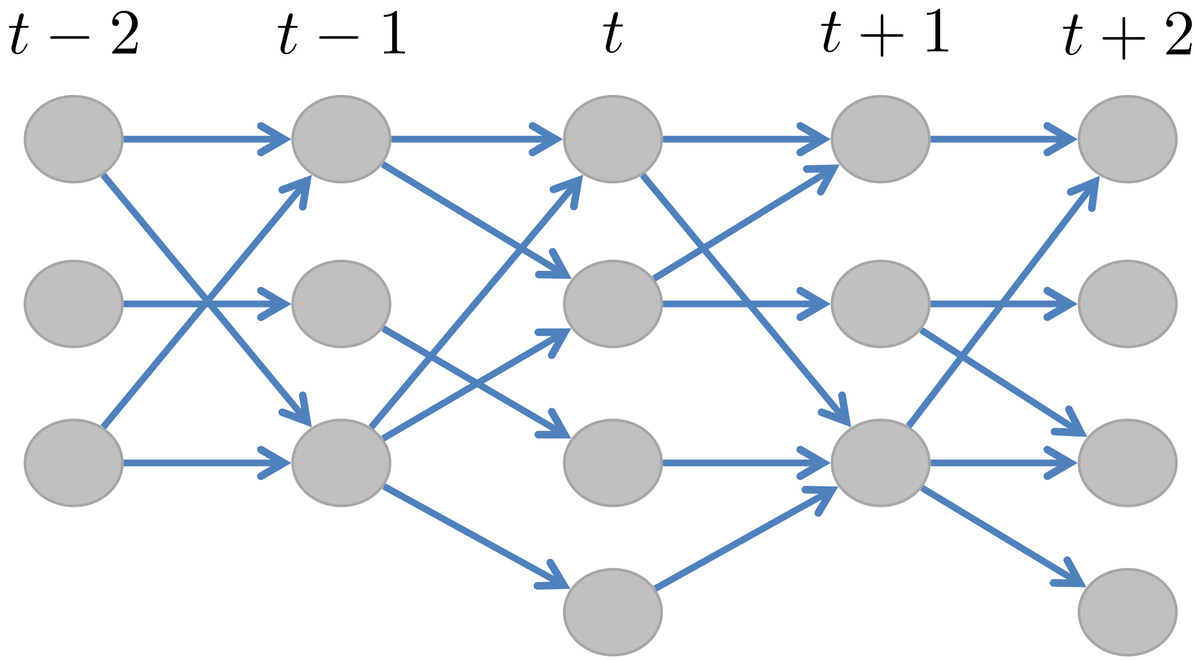
Tracking with network flows

- Minimising the cost:
- Example costs:

$$f^* = \arg\min_f \sum_{i,j} C(i,j) f(i,j)$$

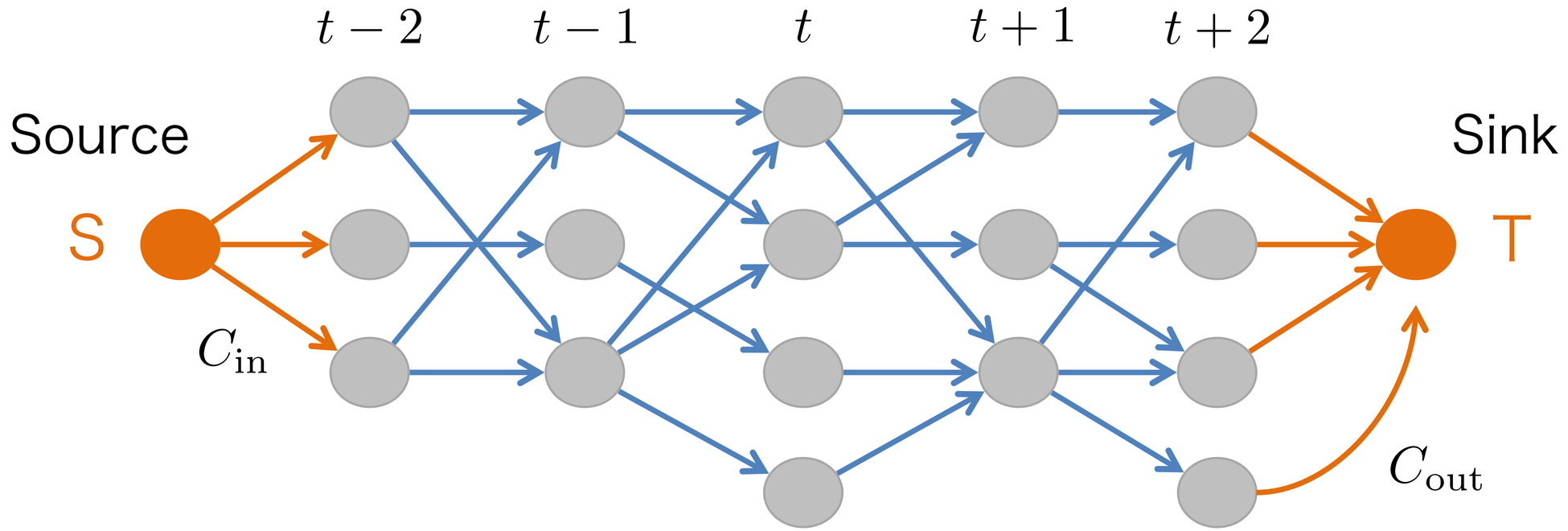


Constructing cost-flow network



C_t transition cost (e.g. using metric learning, motion prior etc.)

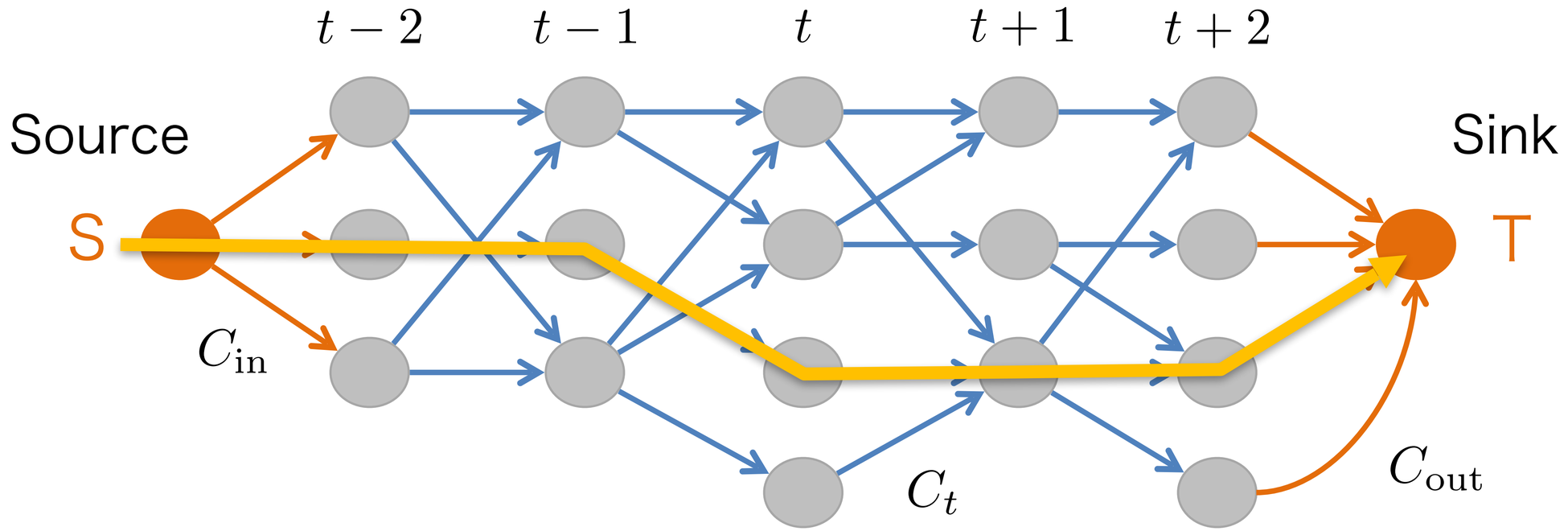
Constructing cost-flow network



C_t transition cost (e.g. using metric learning, motion prior etc.)

C_{in} C_{out} entrance/exit cost: the cost to to start or end a trajectory

Constructing cost-flow network

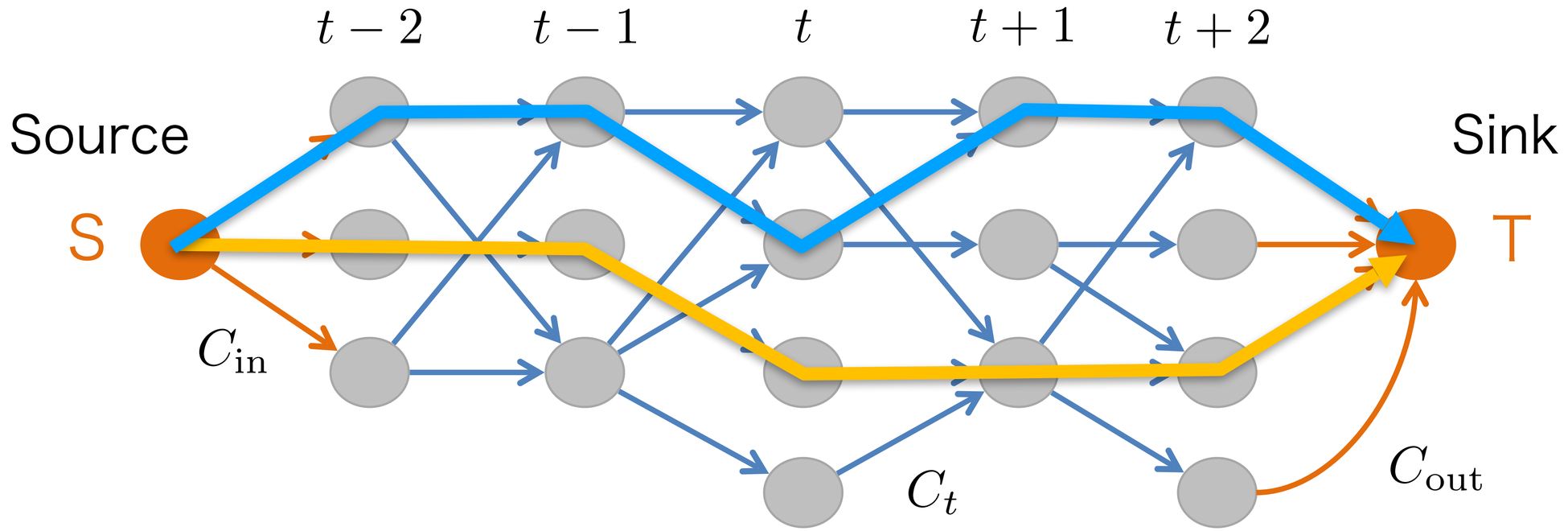


C_t transition cost (e.g. using metric learning, motion prior etc.)

C_{in} C_{out} entrance/exit cost: the cost to to start or end a trajectory

Trajectory: a path starting at S and ending at T.

Constructing cost-flow network



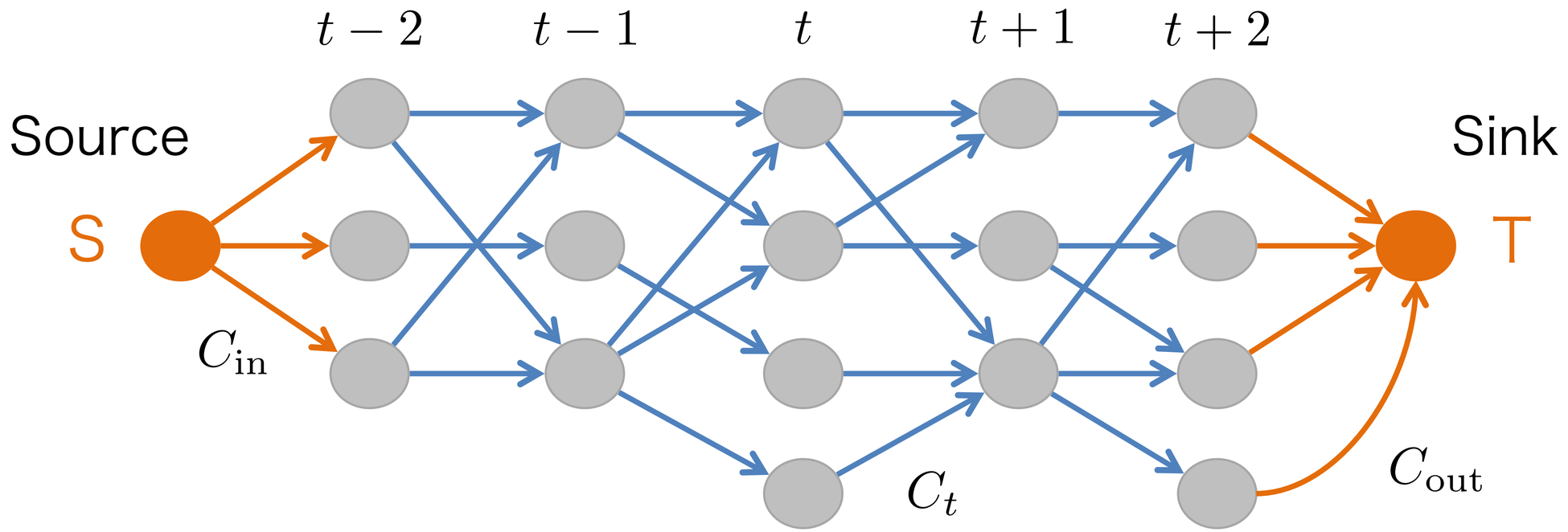
C_t transition cost (e.g. using metric learning, motion prior etc.)

C_{in} C_{out} entrance/exit cost: the cost to to start or end a trajectory

Trajectory: a path starting at S and ending at T.

Trajectories are disjoint ordered sets.

Constructing cost-flow network



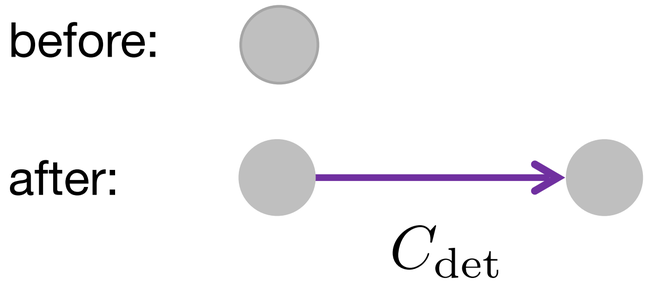
C_t transition cost (e.g. using metric learning, motion prior etc.)

C_{in} C_{out} entrance/exit cost: the cost to to start or end a trajectory

Open question: how to incorporate detection confidence?

Tracking with network flows

- Split the node in two.
- Assign the cost using detection confidence.



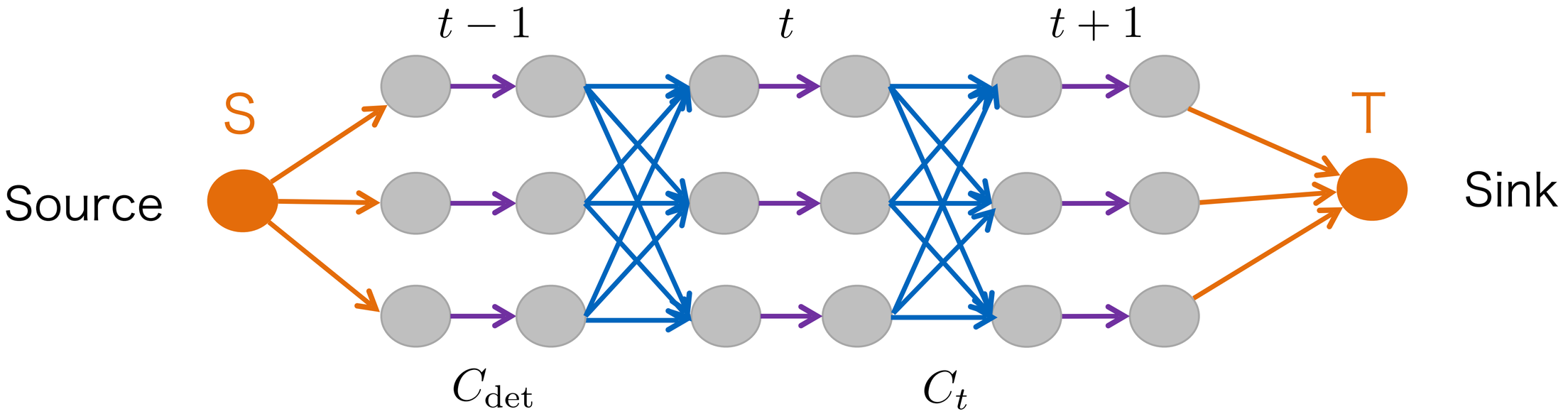
"detection" cost
Important: can be positive or negative

Otherwise, if $C \geq 0$, there is a trivial solution (QUIZ):

$$f^* = \arg\min_f \sum_{i,j} C(i,j) f(i,j)$$

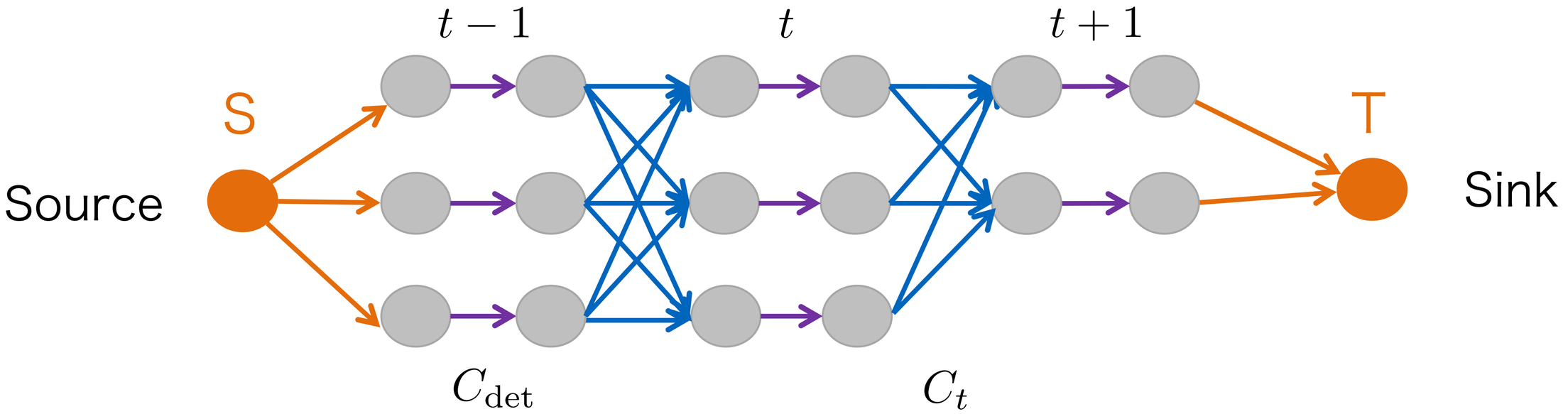
Zhang et al. "Global Data Association for Multi-Object Tracking Using Network Flows". CVPR 2008

Complete graph



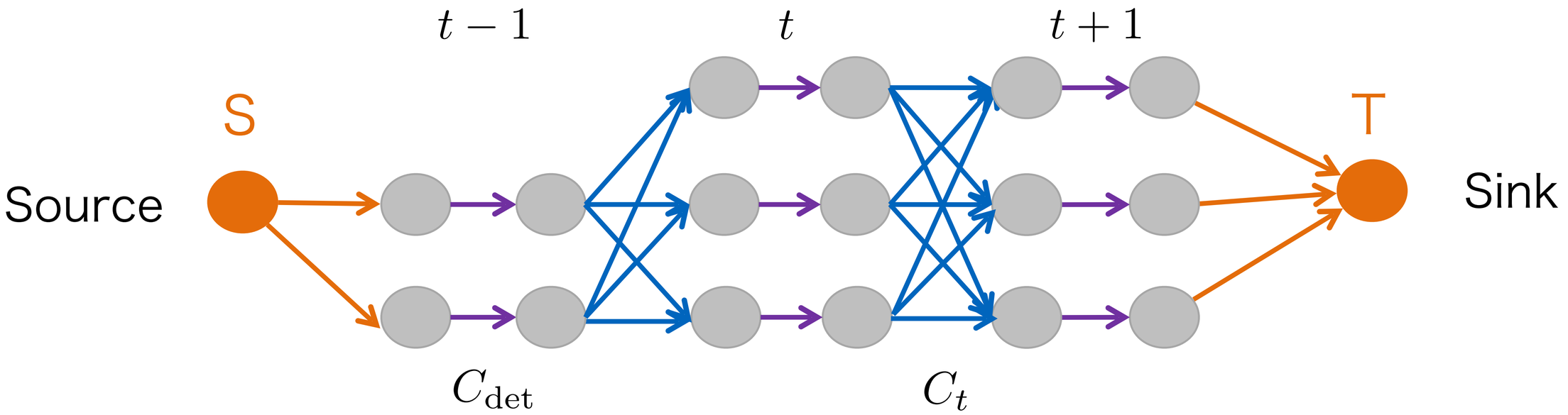
- The graph compactly encodes our problem.
  - QUIZ: How many trajectories (full-length) are possible?
- But... detections are not perfect.

Complete graph



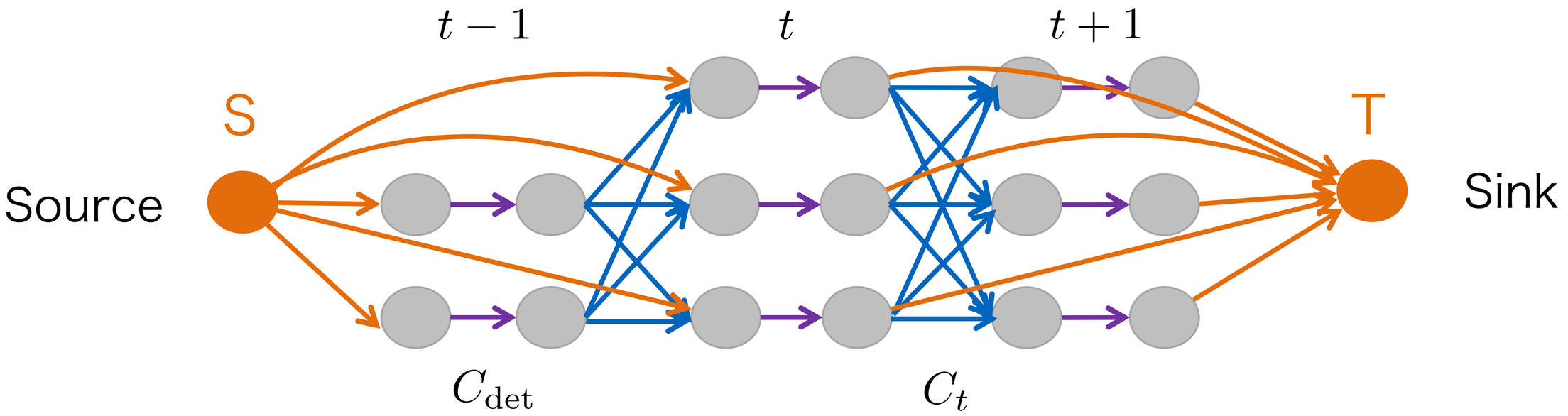
- If there is an occlusion, we find only two trajectories
  - e.g., occlusion in the last frame

Complete graph



- If there is an occlusion, we find only two trajectories
  - or the object appears only in the second frame

Complete graph



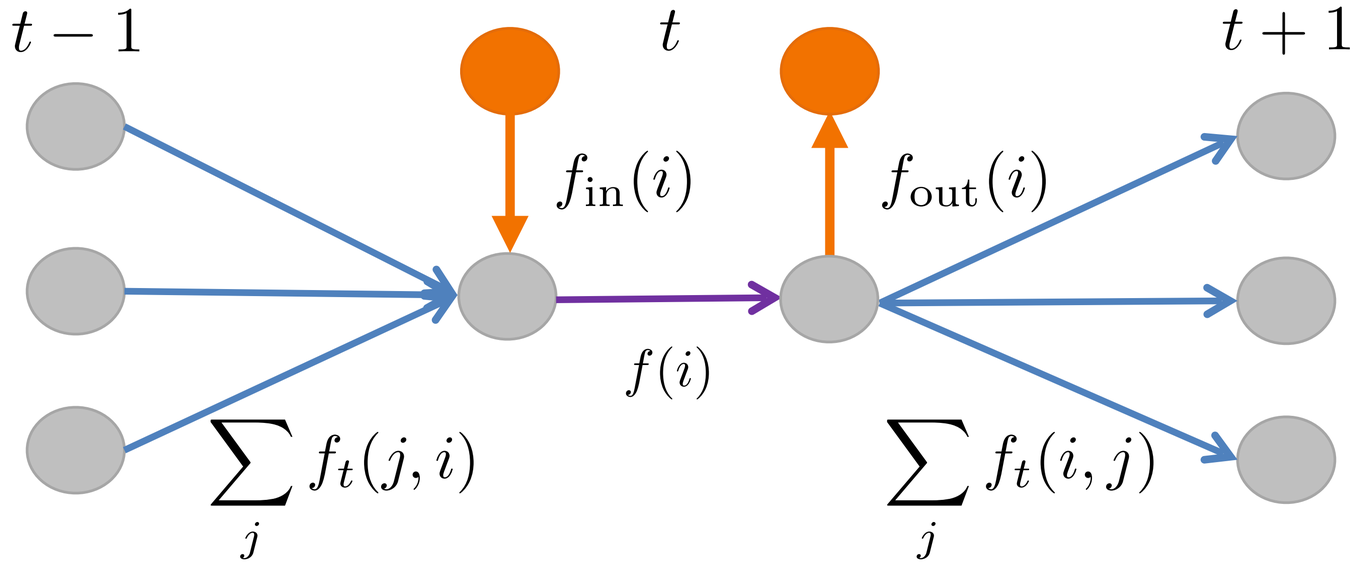
- Solution: Connect <u>all</u> nodes (detections) to entrance/exit nodes:

Constraints

- Subject to **flow conservation**

$$\sum_j f_t(j,i) + f_{\text{in}}(i) = f(i) = \sum_j f_t(i,j) + f_{\text{out}}(i) \qquad \forall f(\cdot,\cdot) \in \{0,1\}$$

$$\forall f(\cdot) \in \{0,1\}$$



MAP formulation

- Our solution is a set of trajectories $\mathcal{T}^*$

$$\mathcal{T}^* = \arg\max_{\mathcal{T}} P(\mathcal{T} \mid \mathcal{X})$$

$$= \arg\max_{\mathcal{T}} P(\mathcal{X}|\mathcal{T})P(\mathcal{T})$$

$$= \arg\max_{\mathcal{T}} \prod_i P(\mathbf{x}_i|\mathcal{T})P(\mathcal{T})$$

$$= \arg\max_{\mathcal{T}} \prod_i P(\mathbf{x}_i|\mathcal{T}) \prod_{T_i \in \mathcal{T}} P(T_i)$$

data likelihood (motion) prior

Bayes rule

Assumption 1:
Conditional independence of observations

Assumption 2:
Independence of trajectories

See Leal-Taixé et al. (2011) for a more advanced model.

Zhang et al. “Global Data Association for Multi-Object Tracking Using Network Flows“. CVPR 2008

MAP formulation

$$\arg\max_{\mathcal{T}} \prod_{i} P(\mathbf{x}_i|\mathcal{T}) \prod_{T_i \in \mathcal{T}} P(T_i)$$

$$= \arg\min_{\mathcal{T}} -\sum_{i} \log P(\mathbf{x}_i|\mathcal{T}) - \sum_{T_i \in \mathcal{T}} \log P(T_i)$$

log-space for optimisation

Prior

$$\sum_{T_i \in \mathcal{T}} \log P(T_i)$$

- Trajectory model: count the entrance, exit and transition costs

$$T_i := \{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_n\}$$

$$P(T_i) = P_{\text{in}}(\mathbf{x_0}) \prod_{j=1,n} P_{\text{t}}(\mathbf{x}_j \mid \mathbf{x}_{j-1}) P_{\text{out}}(\mathbf{x}_n)$$

$$-\log P(T_i) = \boxed{-\log P_{\text{in}}(\mathbf{x_0})} - \sum_{j=1,n} \boxed{\log P_{\text{t}}(\mathbf{x}_j \mid \mathbf{x}_{j-1})} \boxed{-\log P_{\text{out}}(\mathbf{x}_n)}$$

$$f_{\text{in}}(\mathbf{x}_0) C_{\text{in}}(\mathbf{x}_0) \qquad f_t(\mathbf{x}_j, \mathbf{x}_{j-1}) C_t(\mathbf{x}_j, \mathbf{x}_{j-1}) \qquad f_{\text{out}}(\mathbf{x}_n) C_{\text{out}}(\mathbf{x}_n)$$

Likelihood

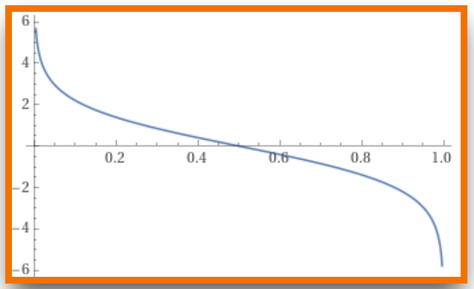
$$-\sum_i \log P(\mathbf{x}_i|\mathcal{T})$$

- We can use Bernoulli distribution:

$$P(\mathbf{x}_i \mid \mathcal{T}) := \begin{cases} \gamma_i, & \text{if } \exists T_j \in \mathcal{T}, \mathbf{x}_i \in T_j \\ 1-\gamma_i, & \text{otherwise} \end{cases}$$

- γ_i denotes prediction confidence (e.g. provided by the detector)

$$-\log P(\mathbf{x}_i \mid \mathcal{T}) = -f(\mathbf{x}_i)\log\gamma_i - (1-f(\mathbf{x}_i))\log(1-\gamma_i)$$



$$= f(\mathbf{x}_i)\underbrace{\log\frac{1-\gamma_i}{\gamma_i}}_{C_{\text{det}}(\mathbf{x}_i)} - \log(1-\gamma_i)$$

can be ignored in optimisation

Optimisation

- $(C_{\text{det}}, C_{\text{in}}, C_{\text{out}}, C_t)$ are estimated from data. Then:

- Construct the graph $G(V, E, C, f)$ from observation set $\mathcal{X}$
- Start with empty flow
- WHILE ($f(G)$ can be augmented)
  - Augment $f(G)$ by one.
  - Find the min cost flow by the algorithm of [12].
  - IF (current min cost $<$ global optimal cost)

    Store current min-cost assignment as global optimum.
- Return the global optimal flow as the best association hypothesis

Binary/Fibonacci search

$O(\log n)$

Min-cost flow algorithm

$O(n^2 m \log n)$

Zhang et al. “Global Data Association for Multi-Object Tracking Using Network Flows“. CVPR 2008

Summary

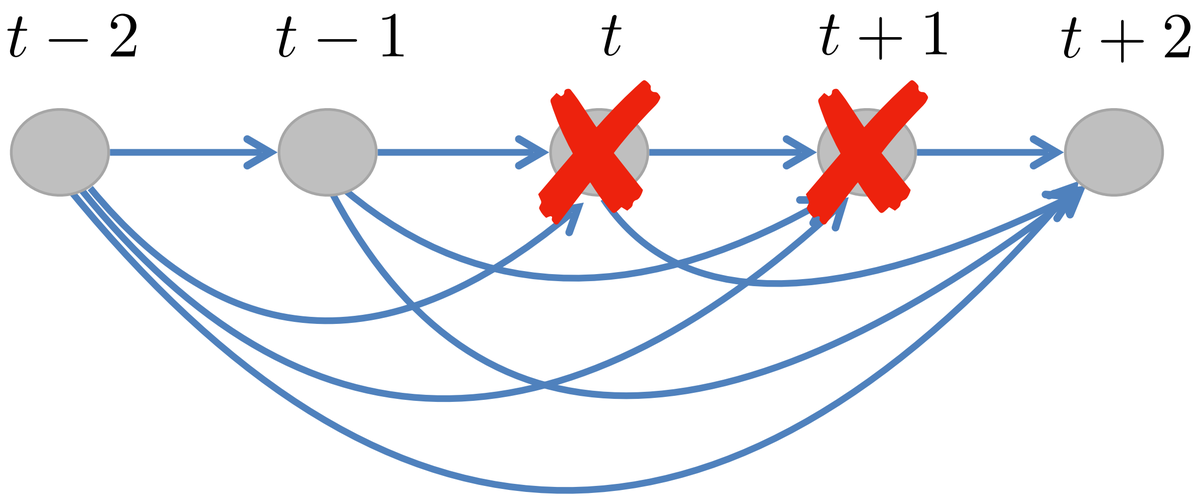
- Min-cost max-flow formulation:
  - the maximum number of trajectories with minimum costs.
- Optimisation maximises maximum a-posteriori (MAP):
  - global solution and efficient (polynomial time).

Open questions

- How to handle occlusions?

Handling occlusions

- How to handle occlusions?
  - “Markovian” formulation may not be sufficient (→ dense graphs)



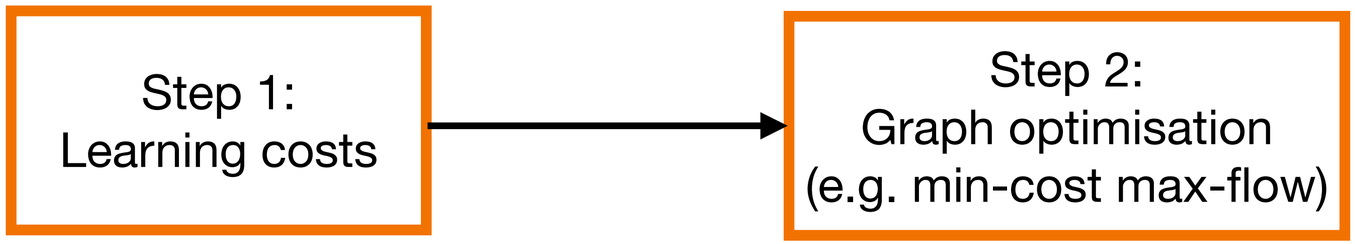
Handling occlusions

More advanced (dense) graph formulations:

- M. Keuper et al. „Motion segmentation and multiple object tracking by correlation co-clustering“ (TPAMI 2018).
- S. Tang et al. „Subgraph decomposition for multi-target tracking” (CVPR 2015).
- S. Tang et al. „Multiple people tracking by lifted multicut and person reidentification“ (CVPR 2017).

Open questions

- How to handle occlusions?
- **How to learn costs?**

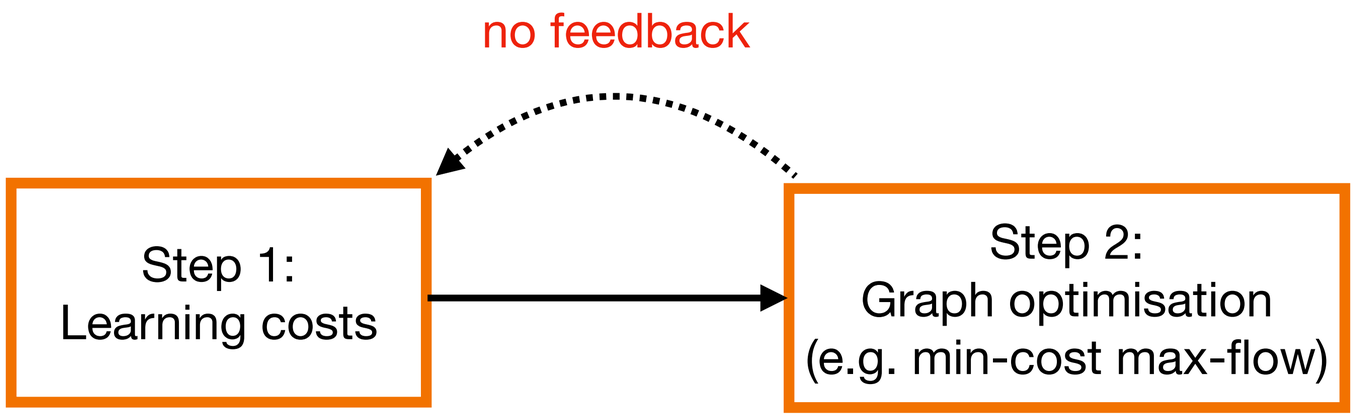


Improving costs

- L. Leal-Taixé et al. “Learning an image-based motion context for multiple people tracking”. CVPR 2014.
- L. Leal-Taixé et al. “Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker“. ICCVW 2011
- L. Leal-Taixé et al. “Learning by tracking: Siamese CNN for robust target association”. CVPRW 2016.
- S. Schulter et al. „Deep network flow for multi-object tracking“. CVPR 2017.
- J. Son at al. „Multi-object tracking with quadruplet convolutional neural networks“. CVPR 2017.
- Ristani and Tomasi. “Features for multi-target multi-camera tracking and re-identification”. CVPR 2018
- J. Xu et al. „Spatial-temporal relation networks for multi-object tracking“. ICCV 2019.

Open questions

- How to handle occlusions?
- **How to learn costs?**
  - costs may be specific to the graph formulation and optimisation



End-to-end learning?

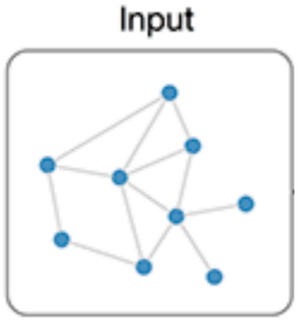
- Can we learn features for multi-object tracking (encoding costs) to encode the solution **directly** on the graph?

- Goal: Generalise the graph structure we have used and perform end-to-end learning

Message Passing Networks

Setup

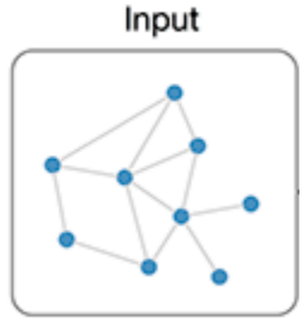
- Input: task-encoding graph
  - nodes: detections encoded as feature vectors
  - edges: node interaction (e.g. inter-frame)



- Output: graph partitioning into (disjoint) trajectories
  - e.g. encoded by edge label (0,1)

Deep learning on graphs

- Key challenges:
  - Graph can be of arbitrary size
    (number of nodes and edges)
  - Need invariance to node permutations



Message Passing Networks

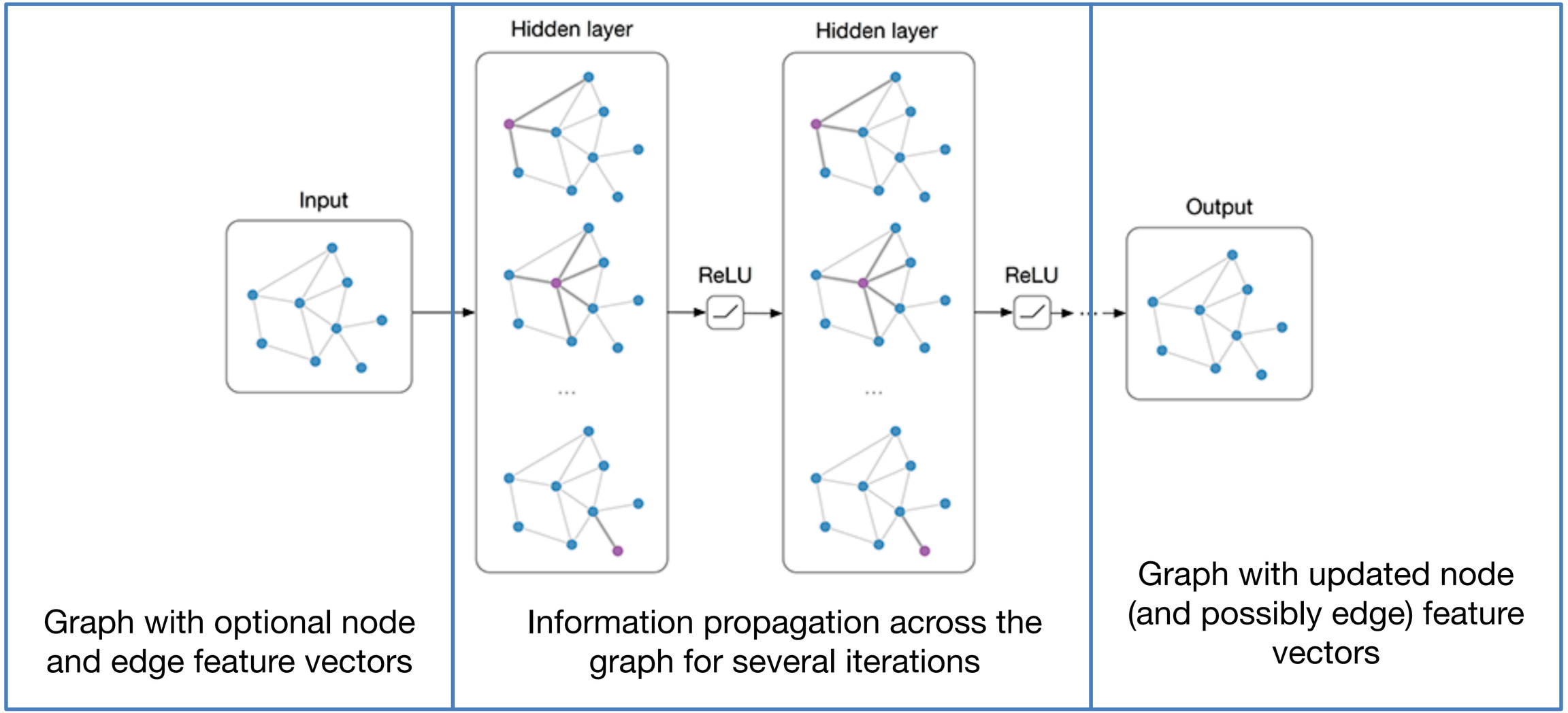


Figure credit: https://tkipf.github.io/graph-convolutional-networks/

Message Passing Networks

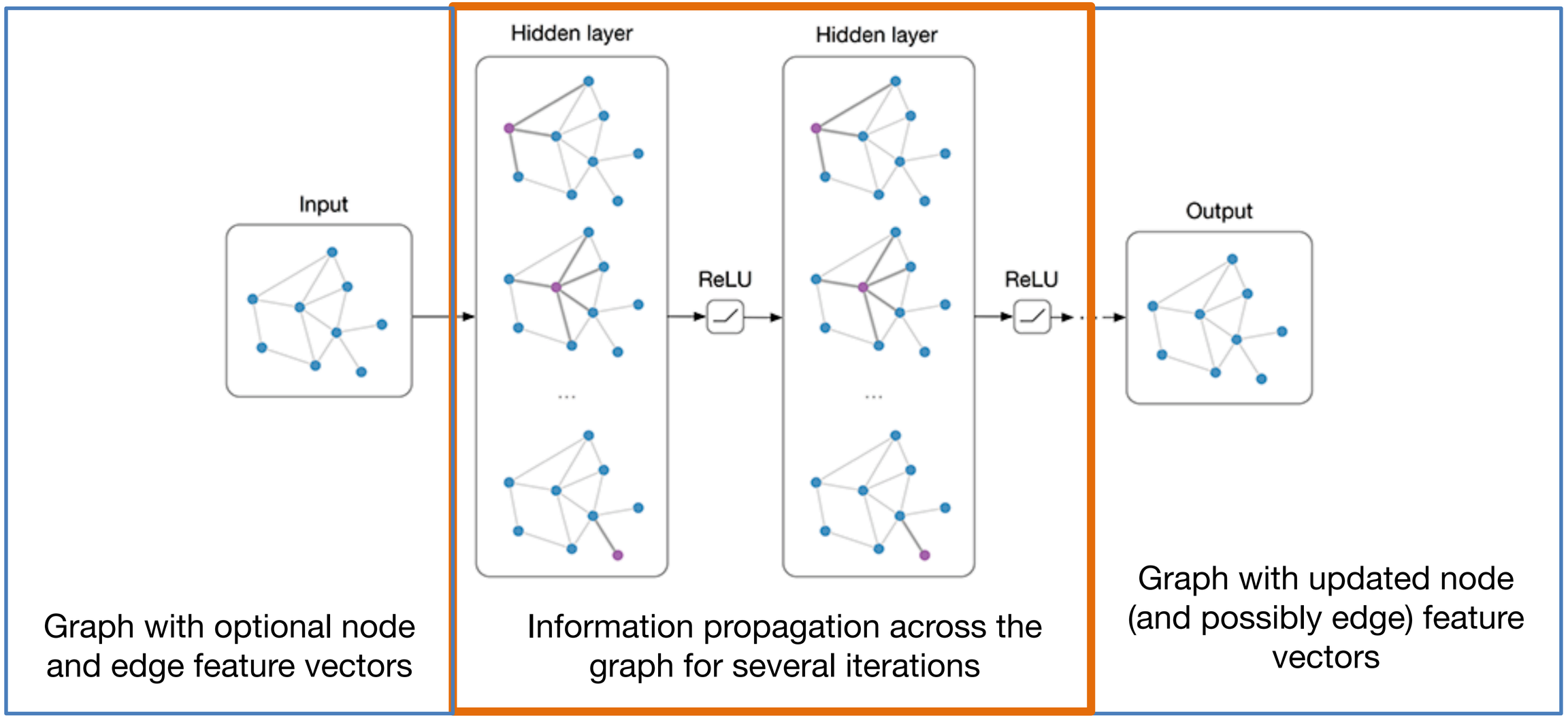
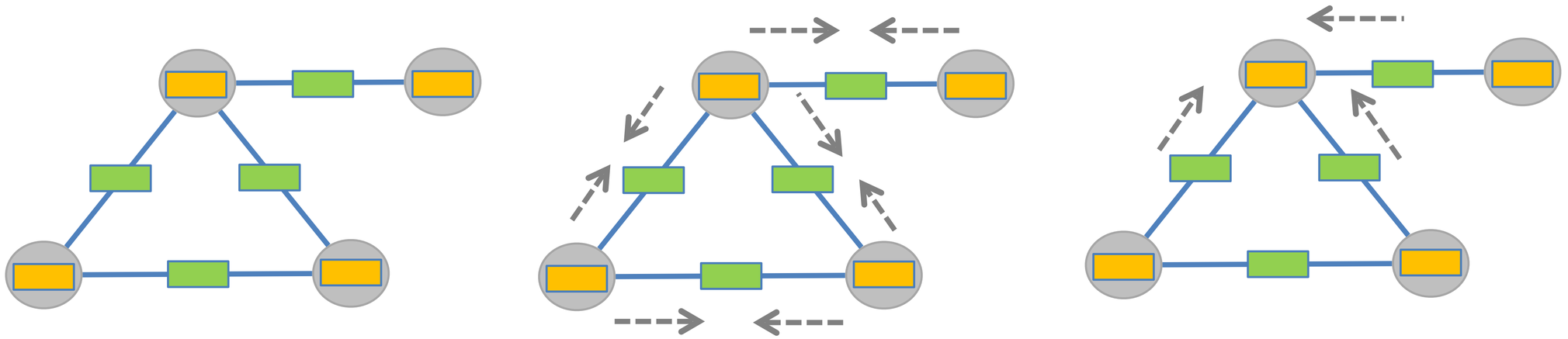


Figure credit: https://tkipf.github.io/graph-convolutional-networks/

Message passing



Initial graph

‘Node-to-edge’ update

‘Edge-to-node’ update

- We can divide the propagation process in two steps:
  - ‘node-to-edge’ and ‘edge-to-node’ updates.
- Alternate these two updates (message passing)

  → encodes context into embeddings.

Node embeddings

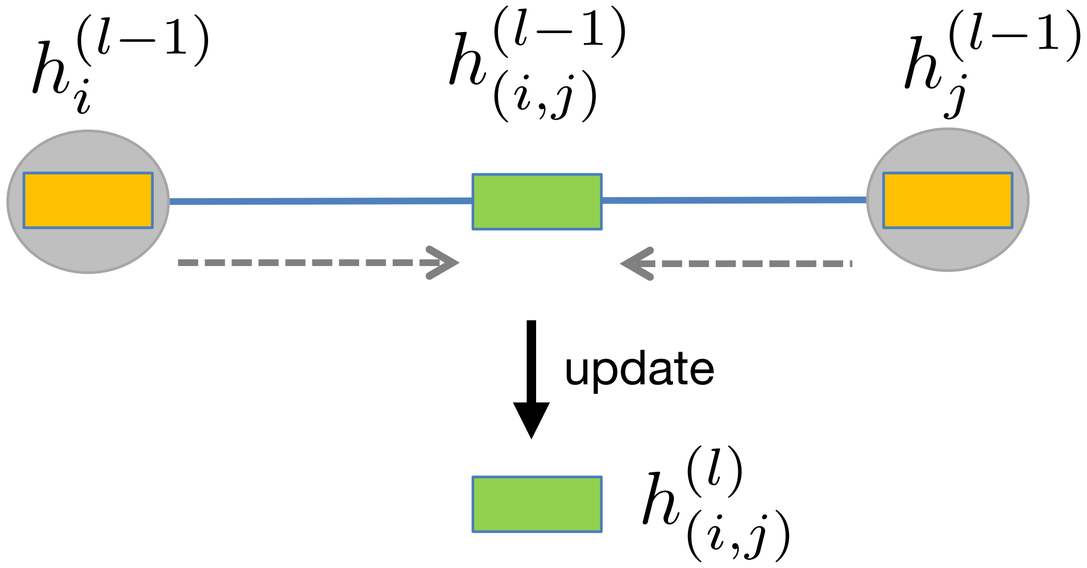
Edge embeddings

Notation

- Graph: $G = (V, E)$
- Initial embeddings:
  - Node embedding: $h_i^{(0)},\ i \in V$
  - Edge embedding: $h_{(i,j)}^{(0)},\ (i,j) \in E$

- Embeddings after l steps: $h_i^{(l)},\ i \in V$ $\qquad$ $h_{(i,j)}^{(l)},\ (i,j) \in E$

Node-to-edge update

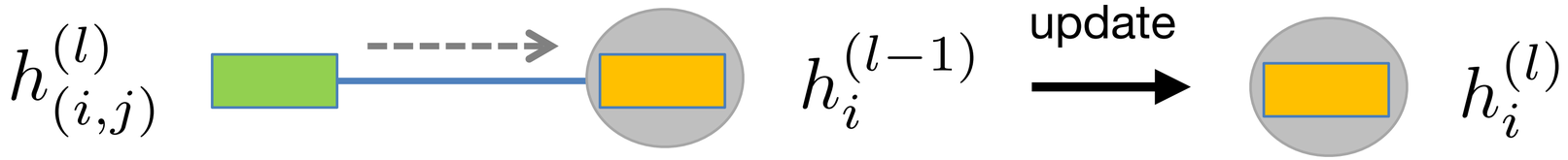


$$h_{(i,j)}^{(l)} = \mathcal{N}_e\left([h_i^{(l-1)}, h_{(i,j)}^{(l-1)}, h_j^{(l-1)}]\right)$$

Learnable function (e.g. MLP)
shared across the entire graph

Edge-to-node updates

- Use the updated edge embeddings to update nodes:



- After a round of edge updates, each edge embedding contains information about its pair of incident nodes.

- By analogy: $h_i^{(l)} = \mathcal{N}_v\left([h_i^{(l-1)}, h_{(i,j)}^{l}]\right)$
- In general, we may have an arbitrary number neighbours (“degree”, or “valency”)

Edge-to-node updates

- Define a permutation-invariant aggregation function:

$$\Phi^{(l)}(i) := \Phi\left(\left\{h^{(l)}(i,j)\right\}_{j \in Ne(i)}\right)$$

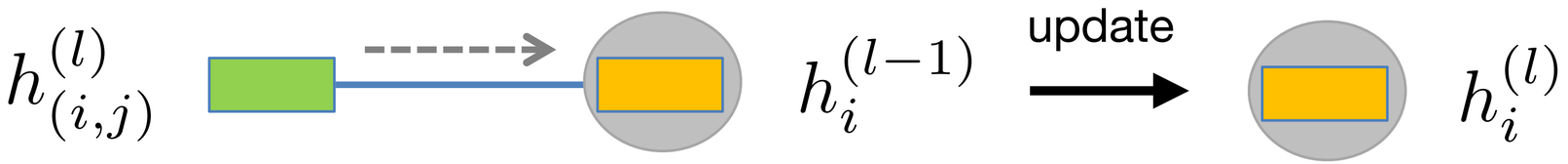
The input is a **set** of embeddings
from incident edges

- What are examples permutation-invariant functions? (QUIZ)

Qi et al., “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation” (2017)

Edge-to-node updates

- Re-define the edge-to-node updates for a general graph:



$$h_i^{(l)} = \mathcal{N}_v\left(h_i^{(l-1)}, \Phi^{(l)}(i)\right)$$

previous state

context information from neighbours

Remarks

- Main goal: gather context information into node and edge embeddings
- Is one iteration of node-to-edge/edge-to-node updates enough? (QUIZ)
- One iteration increases the receptive field of a node/edge by 1
  - in practice: iterate message passing multiple times (hyperparameter).

- All operations used are differentiable!
- All vertices/edges are treated equally, i.e. the parameters a shared.

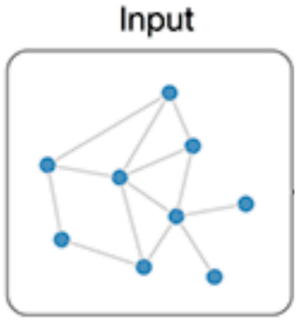
Additional reading

- Several works have proposed generalizations of Neural Networks that can operate on graph-structured domains:
  - Scarselli et al. “The Graph Neural Network Model”. IEEE Trans. Neur. Net 2009.
  - Kipf et al. “Semi-Supervised Classification with Graph Convolutional Networks. ICLR 2016.
  - Gilmer et al. “Neural Message Passing for Quantum Chemistry”. ICML 2017
  - Battaglia et al. “Relational inductive biases, deep learning, and graph networks”. arXiv 2018 (review paper)

MOT
with Message Passing Networks

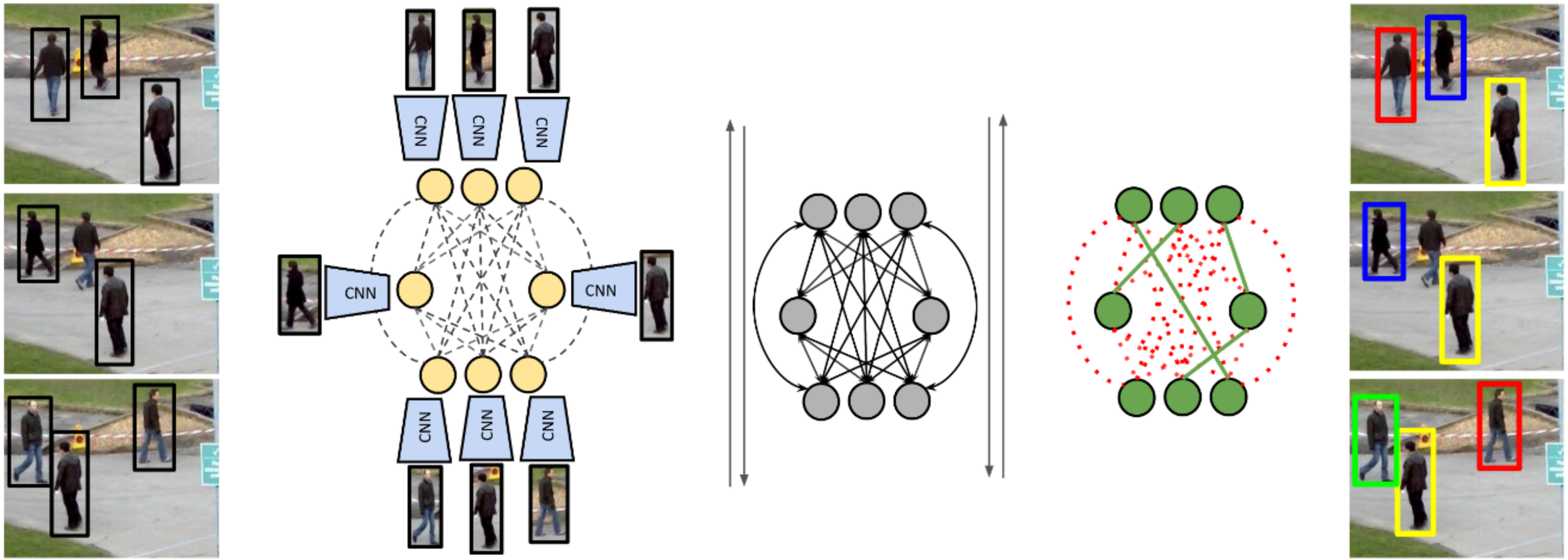
Recall our setup

- Input: task-encoding graph
  - nodes: detections encoded as feature vectors
  - edges: node interaction (e.g. inter-frame)



- Output: graph partitioning into (disjoint) trajectories
  - e.g. encoded by edge label (0,1)

Overview

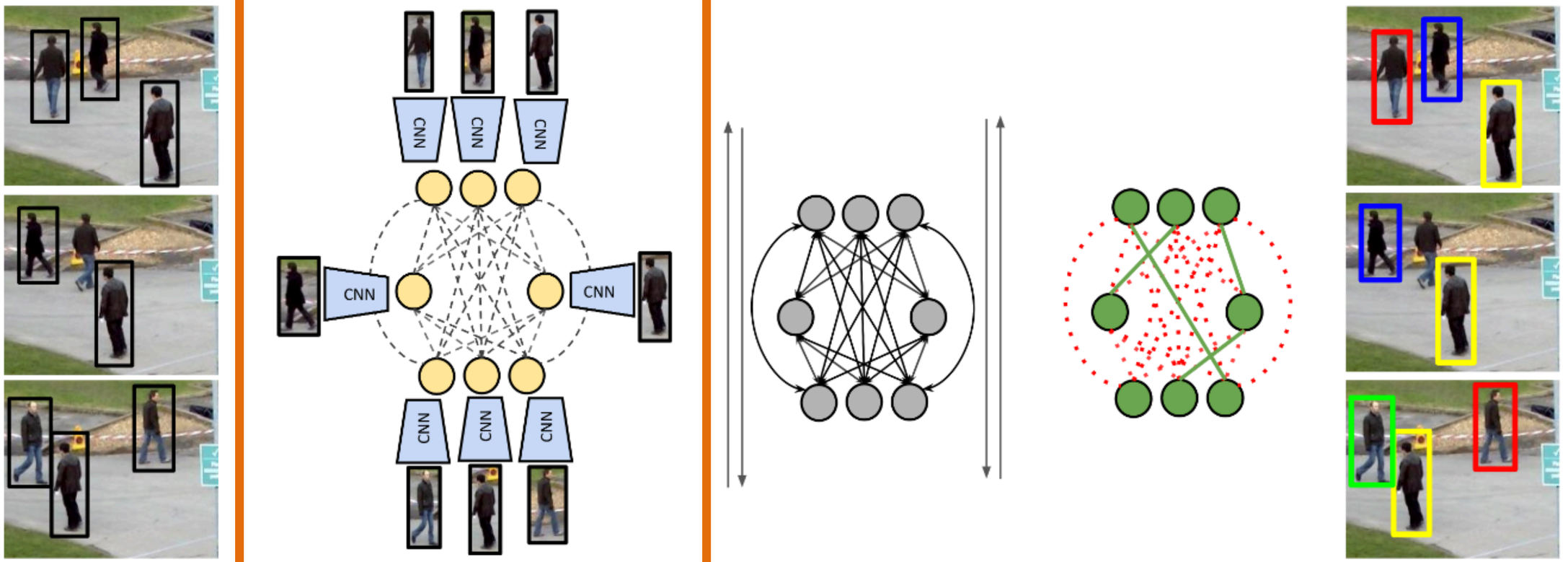


(a) Input (b) Graph Construction + Feature Encoding (c) Neural Message Passing (d) Edge Classification (e) Output

Brasó and Leal-Taixé. “Learning a Neural Solver for Multiple Object Tracking” (2019).

Overview

Encode appearance and scene geometry cues into node and edge embeddings

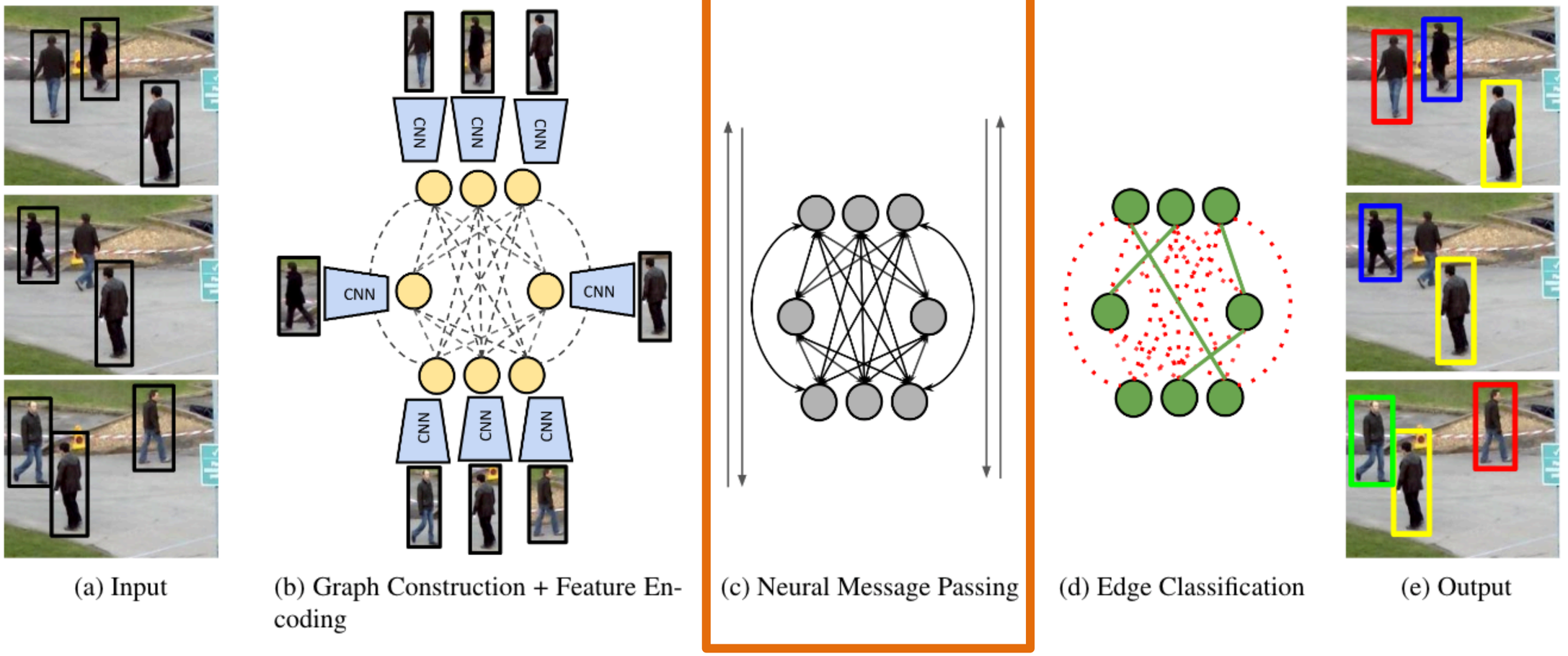


(a) Input (b) Graph Construction + Feature Encoding (c) Neural Message Passing (d) Edge Classification (e) Output

Brasó and Leal-Taixé. “Learning a Neural Solver for Multiple Object Tracking” (2019).

Overview

Propagate cues across the entire graph
with neural message passing

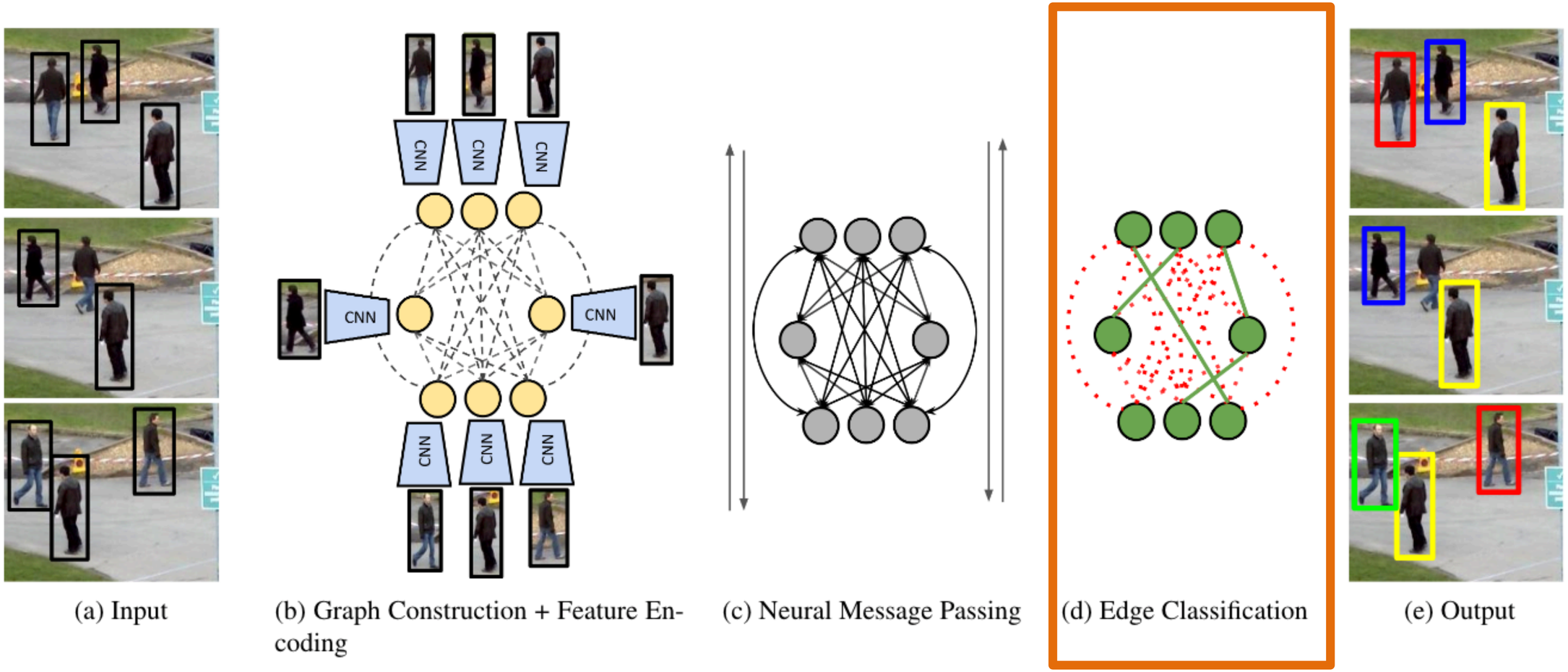


(a) Input (b) Graph Construction + Feature Encoding (c) Neural Message Passing (d) Edge Classification (e) Output

Brasó and Leal-Taixé. “Learning a Neural Solver for Multiple Object Tracking” (2019).

Overview

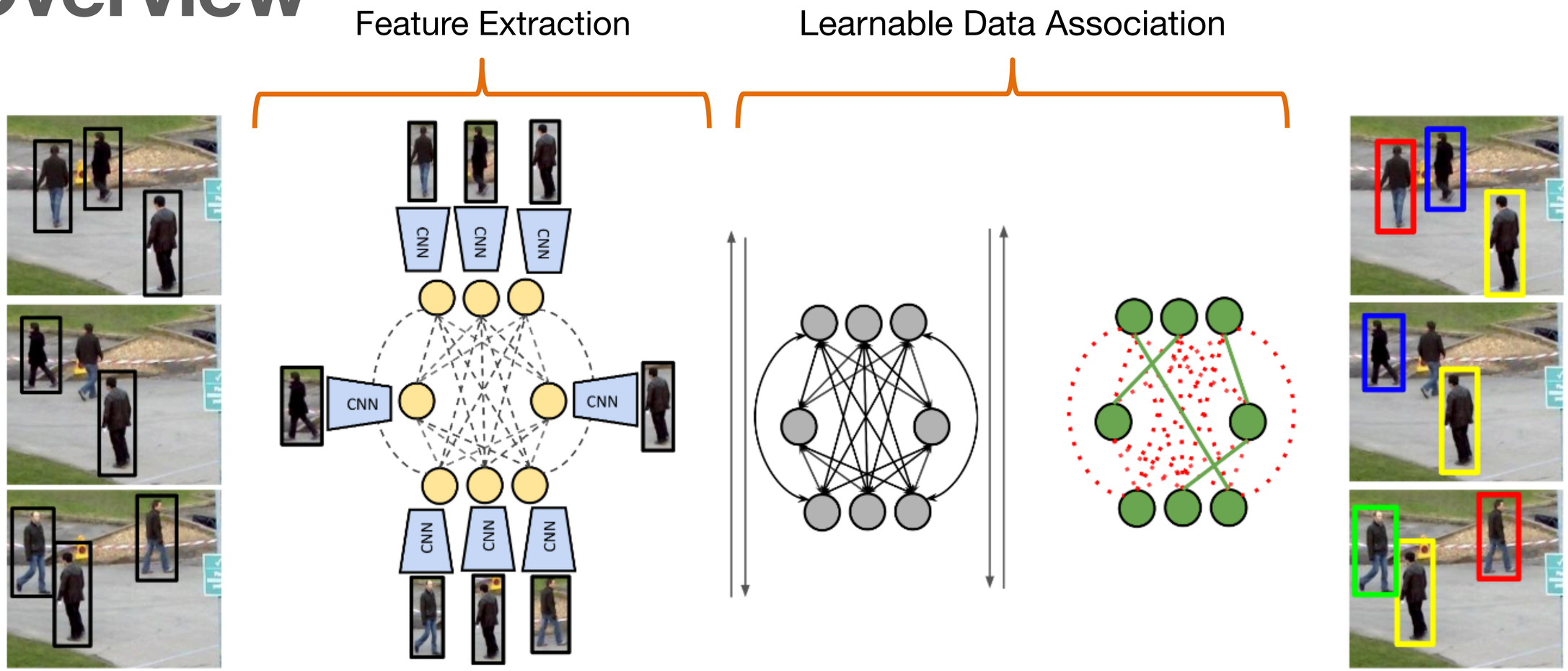
Learn to directly predict solutions of the Min-Cost Flow problem by classifying edge embeddings



(a) Input (b) Graph Construction + Feature Encoding (c) Neural Message Passing (d) Edge Classification (e) Output

Brasó and Leal-Taixé. “Learning a Neural Solver for Multiple Object Tracking” (2019).

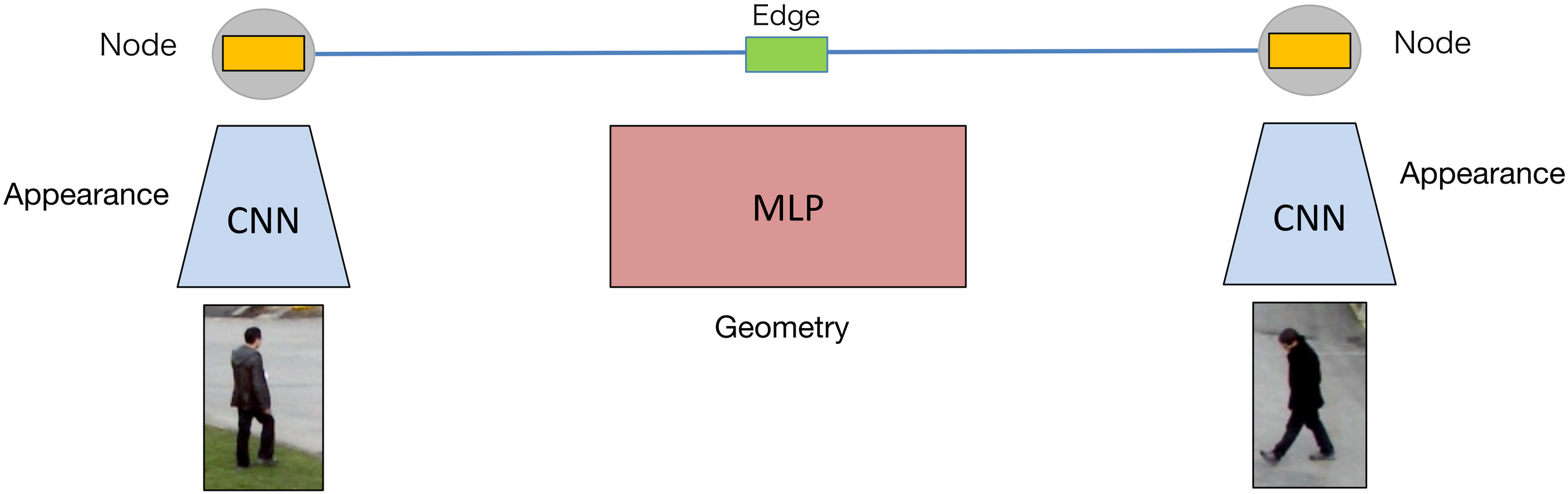
Overview



Brasó and Leal-Taixé. “Learning a Neural Solver for Multiple Object Tracking” (2019).

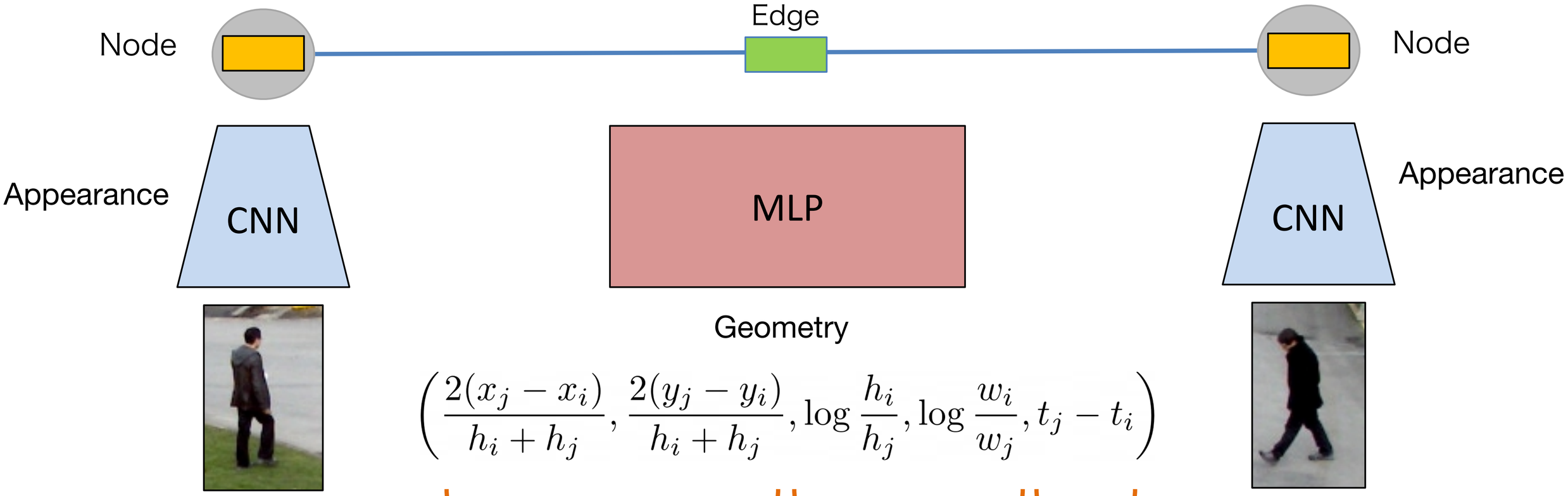
Feature encoding

- Appearance and geometry encodings



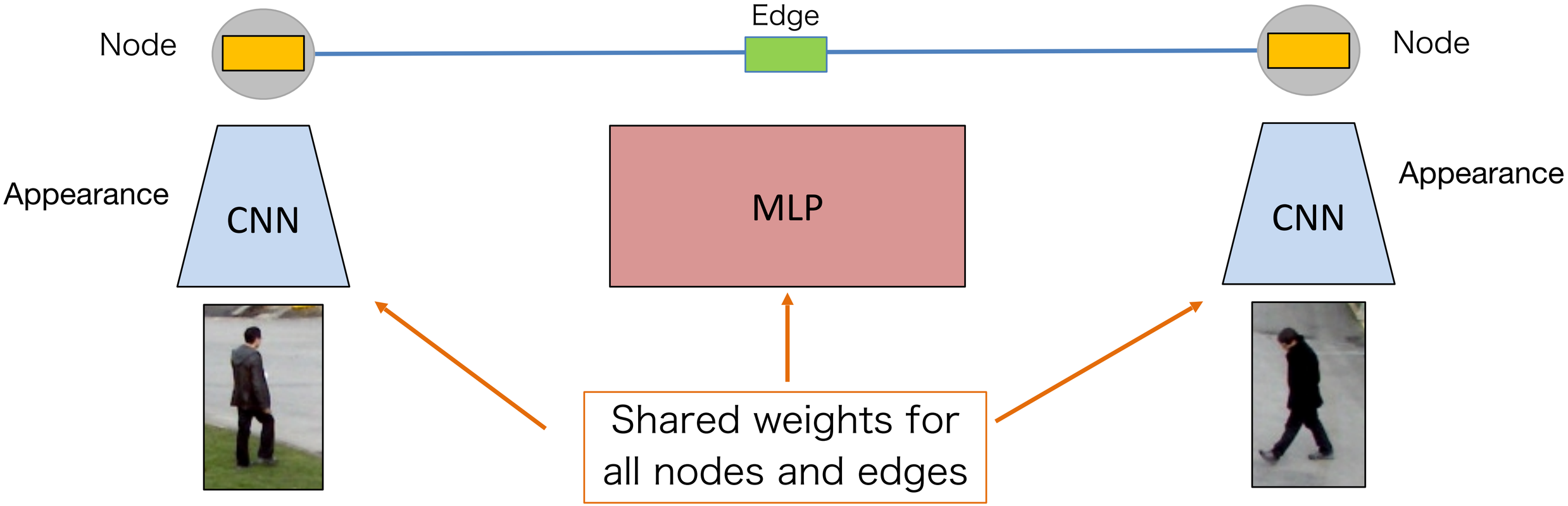
Feature encoding

- Appearance and geometry encodings



Feature encoding

- Appearance and geometry encodings

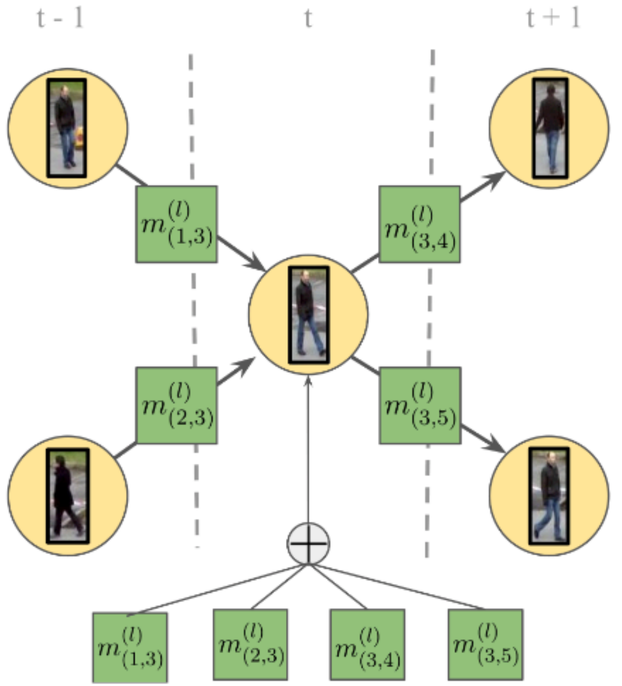


Feature encoding

- Contrast:
  - earlier: defining pairwise and unary costs
  - now:
    - feature vectors associated to nodes and edges
    - use message passing to aggregate context information into the features

Temporal causality

- Flow conservation at a node:
  - at most 1 connection to past
  - at most 1 connection to future

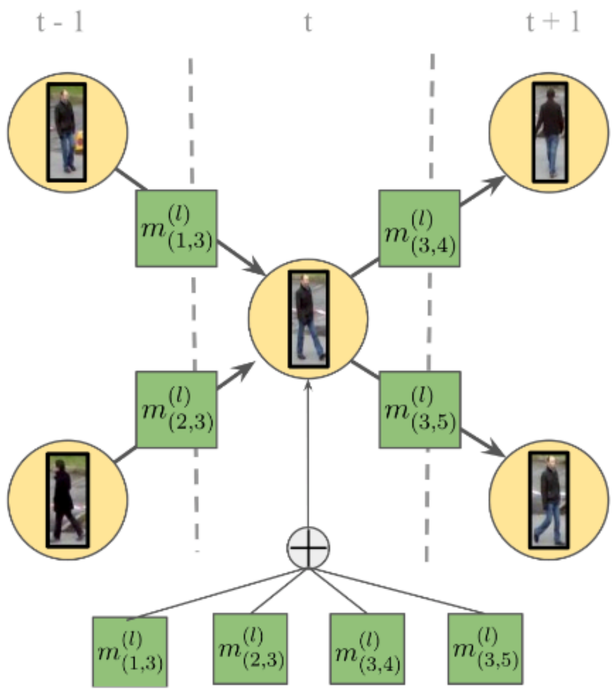


- Is this edge-to-node aggregation suitable?:

$$\Phi^{(l)}(i) := \Phi\left(\left\{h^{(l)}(i,j)\right\}_{j \in Ne(i)}\right)$$

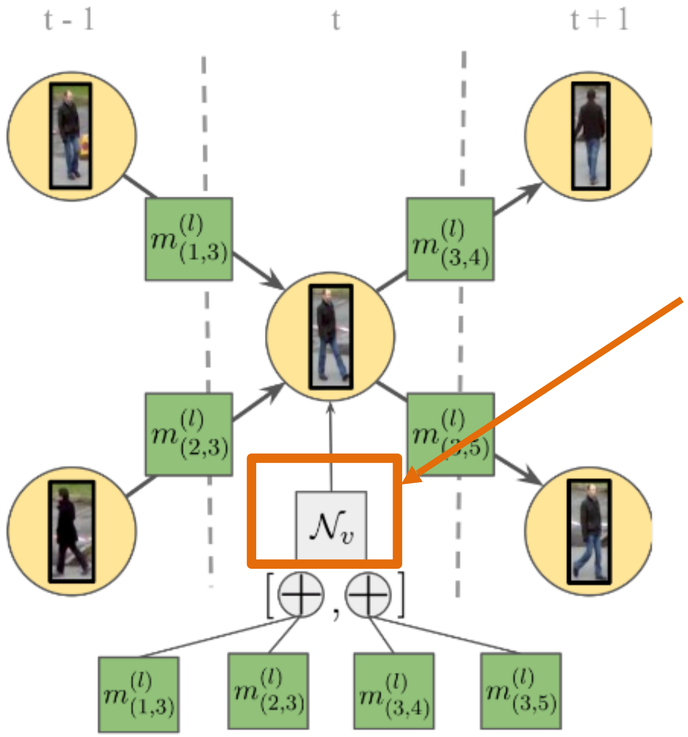
- Solution: decouple incident from outgoing edges

Time-aware Message Passing



(b) Vanilla node update

All node embeddings are aggregated at once



(c) Time-aware node update

An additional network combines both sources of aggregated features

Aggregation of nodes is separated between past / future frames

Brasó and Leal-Taixé. “Learning a Neural Solver for Multiple Object Tracking” (2019).

Classifying edges

- After several iterations of message passing, each edge embedding contains context information about detections
- We feed the embeddings to an MLP that predicts whether an edge is active/inactive

$$\mathcal{L} = \frac{-1}{|E|} \sum_{l=l_0}^{l=L} \sum_{(i,j) \in E} w \cdot y_{(i,j)} \log(\hat{y}_{(i,j)}^{(l)}) + (1 - y_{(i,j)}) \log(1 - \hat{y}_{(i,j)}^{(l)})$$

Sum over the last steps

Weight to balance active / inactive edges

Edge predictions (w. sigmoid) at iteration l

Binary cross-entropy

Obtaining final solutions

- After classifying edges, we get a prediction between 0 and 1 for each edge in the graph.
- Directly thresholding solutions could yield infeasible solutions (e.g. violating flow conservation constraints)
- Lightweight post-processing can be used to obtain final trajectories
  - (e.g. linear programming, see Brasó and Leal-Taixé (2019))
- In practice, around 98% of constraints are automatically satisfied, and rounding takes negligible time
- The overall method is fast (~12 fps) and achieves SOTA in the MOT Challenge by a significant margin

Summary

- No strong assumptions on the graph structure
  - handling occlusions
- Costs can be can learned from data
- Accurate and fast (for an offline tracker).
- (Almost) End-to-end learning approach
  - some post-processing required

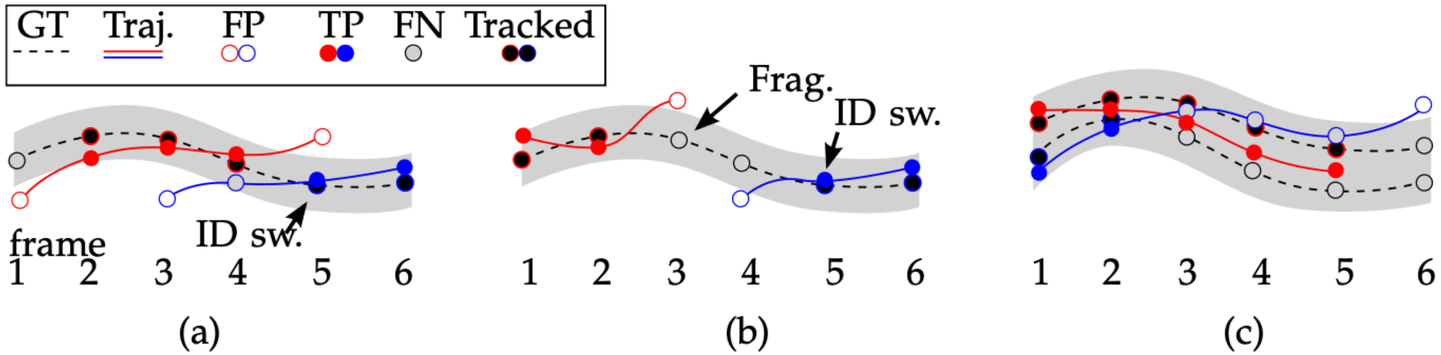
MOT evaluation

Evaluation metrics

- Compute a set of measures per frame
  - Perform matching between predictions and ground truth (we will use exactly the same Hungarian algorithm)
  - FP = false positives
  - FN = false negatives
  - IDSW: “identity switches”

Evaluation metrics

- How do we compute ID switches?



(a) An ID switch is counted because the ground truth track is assigned first to red, then to blue.

(b) Count both an ID switch (red and blue both assigned to the same ground truth), but also a fragmentation (Frag.) because the ground truth coverage was cut.

(c) Identity is preserved. If two trajectories overlap with a ground truth trajectory (within a threshold), the one that forces least ID switches is chosen (the red one).

Leal-Taixé et al.. MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking (2015).

Evaluation metrics

- Compute a set of measures per frame
  - Perform matching between predictions and ground truth (we will use exactly the same Hungarian algorithm)
  - FP = False positives
  - FN = False negatives (missing detections)
  - IDSW: “identity switches”

Multi-object tracking accuracy

$$\mathrm{MOTA} = 1 - \frac{\sum_t (\mathrm{FN}_t + \mathrm{FP}_t + \mathrm{IDSW}_t)}{\sum_t \mathrm{GT}_t},$$

Ground truth

Evaluation metrics

- Multi-object tracking accuracy (MOTA):

$$\mathrm{MOTA} = 1 - \frac{\sum_t (\mathrm{FN}_t + \mathrm{FP}_t + \mathrm{IDSW}_t)}{\sum_t \mathrm{GT}_t},$$

Ground truth

- F_1-Score: $$\mathrm{IDF1} = \frac{2\sum_t \mathrm{TP}_t}{\sum_t 2\mathrm{TP}_t + \mathrm{FP}_t + \mathrm{FN}_t}$$

- Multi-object tracking precision (MOTP): $$\mathrm{MOTP} = \frac{\sum_{t,i} \mathrm{IoU}_{t,i}}{\sum_t \mathrm{TP}}$$

Datasets

- MOTChallenge: www.motchallenge.net (people)
- KITTI benchmark: http://www.cvlibs.net/datasets/kitti/ (vehicles)
- UA-Detrac: http://detrac-db.rit.albany.edu (vehicles)

- Each benchmark may have a unique set challenges
- Unbiased view:
  - high/competitive accuracy on all datasets is desirable