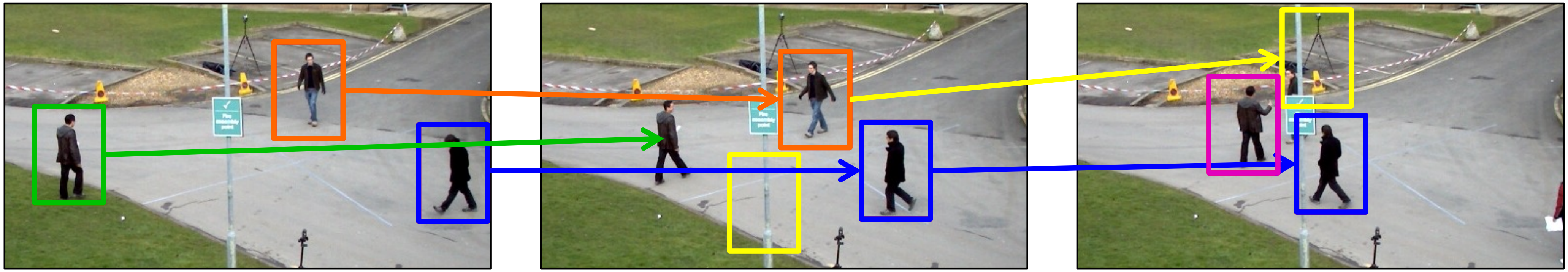
Multiple object tracking

Different challenges

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

Tracking-by-detection

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



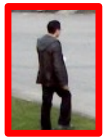
Find detections that match and form a trajectory

Online vs offline tracking

- Online tracking
  - Often 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

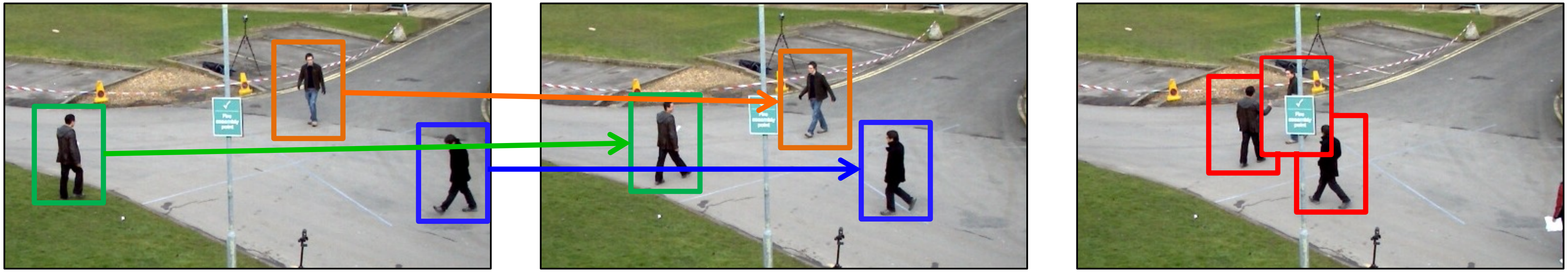
Frame-by-frame

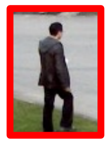
t t+1 t+2

- 1. Track **initialization** (e.g. using a detector) 

Frame-by-frame

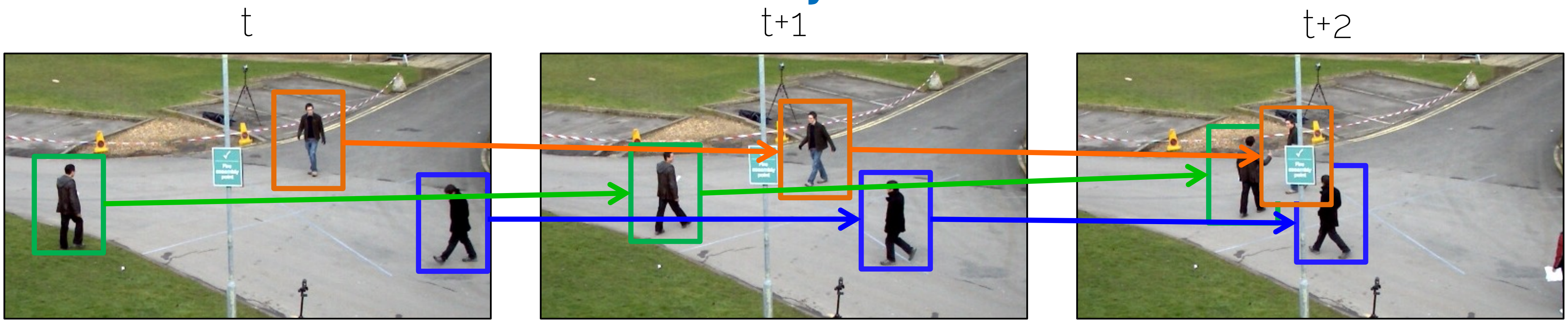
t t+1 t+2

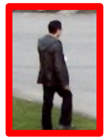


- 1. Track initialization (e.g. using a detector) 

- 2. **Matching** detections at t with detections at t+1

Frame-by-frame



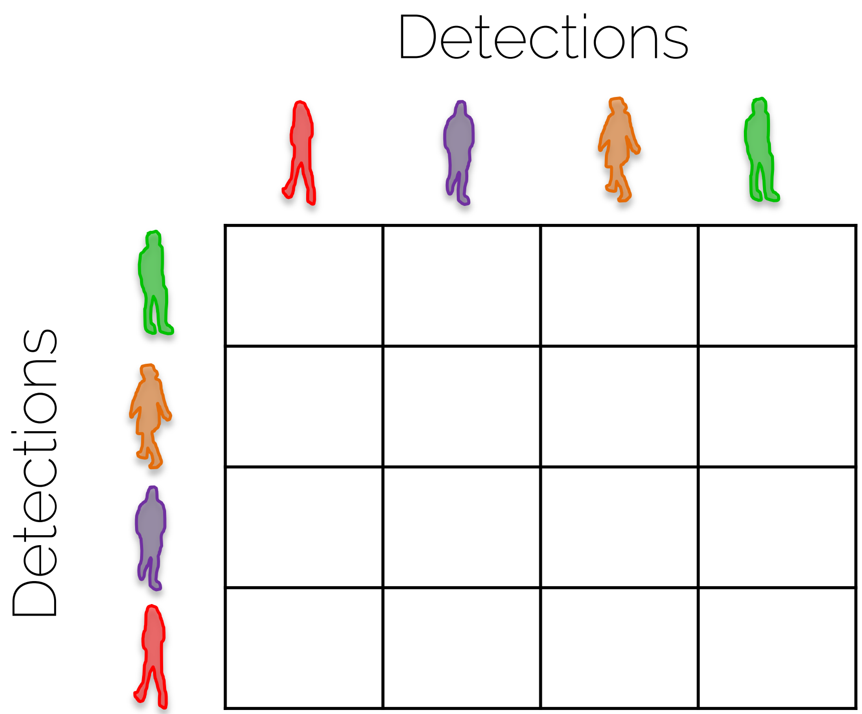
- 1. Track initialization (e.g. using a detector) 

- 2. **Matching** detections at t with detections at t+1

- Repeat for every pair of frames

Frame-by-frame

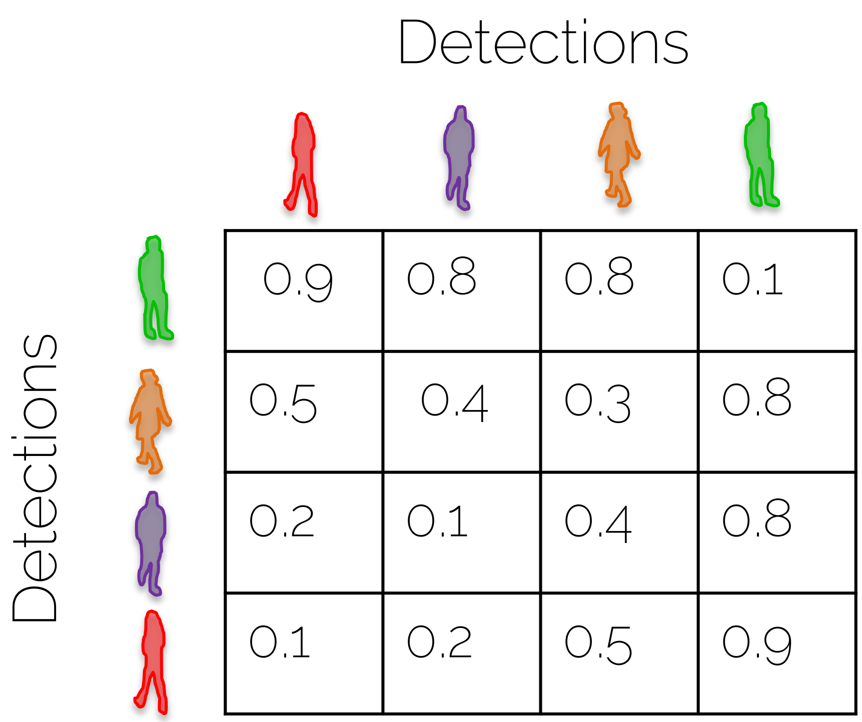
- 3. Matching tracks at t with detections at t+1



Frame-by-frame

- Bipartite matching
  - Define distances between boxes

    (e.g., IoU, pixel distance, 3D distance)



Frame-by-frame

- Bipartite matching
  - Define distances between boxes

    (e.g., IoU, pixel distance, 3D distance)
  - Solve the unique matching with e.g., the Hungarian algorithm*

Detections

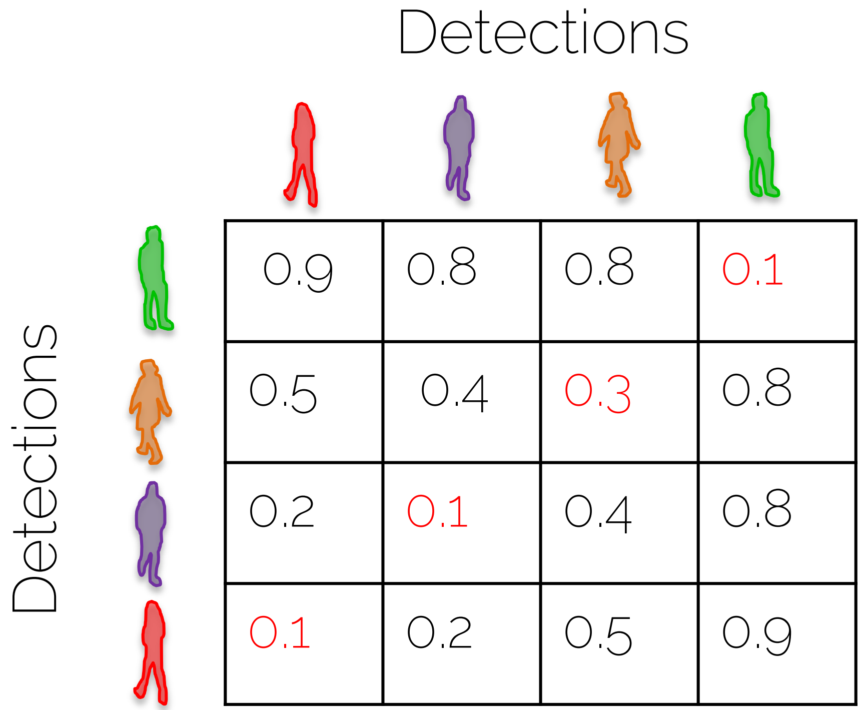
Detections

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

Frame-by-frame

- Bipartite matching
  - Define distances between boxes
    (e.g., IoU, pixel distance, 3D distance)
  - Solve the unique matching with e.g., the Hungarian algorithm*
  - Solutions are the unique assignments that minimize the total cost



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

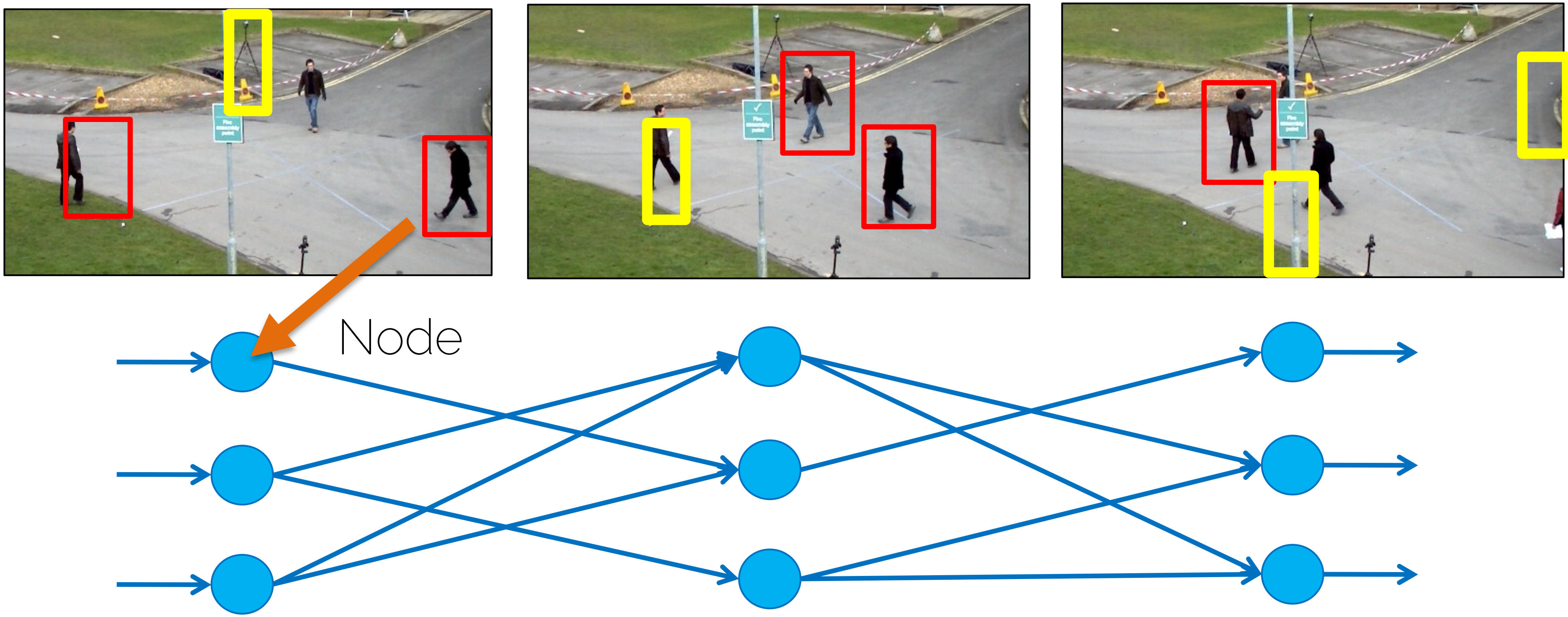
Frame-by-frame

- Problems with frame-by-frame tracking
  - Cannot recover from errors. If a detection is missing from a frame, we have to end the trajectory.
  - All decisions are essentially local
  - Hard to recover from errors in the detection step

- Solution: find the minimum cost solution for ALL frames and ALL trajectories

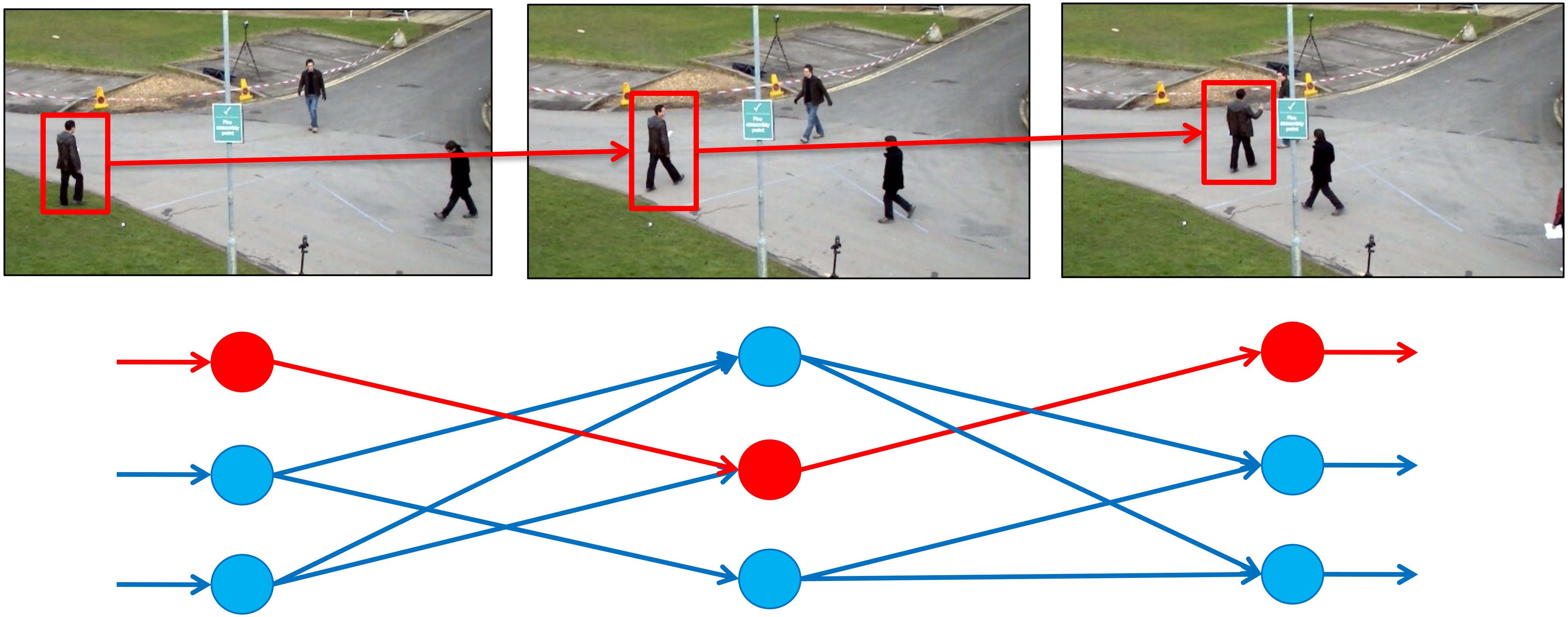
Graph-based MOT

Tracking with network flows



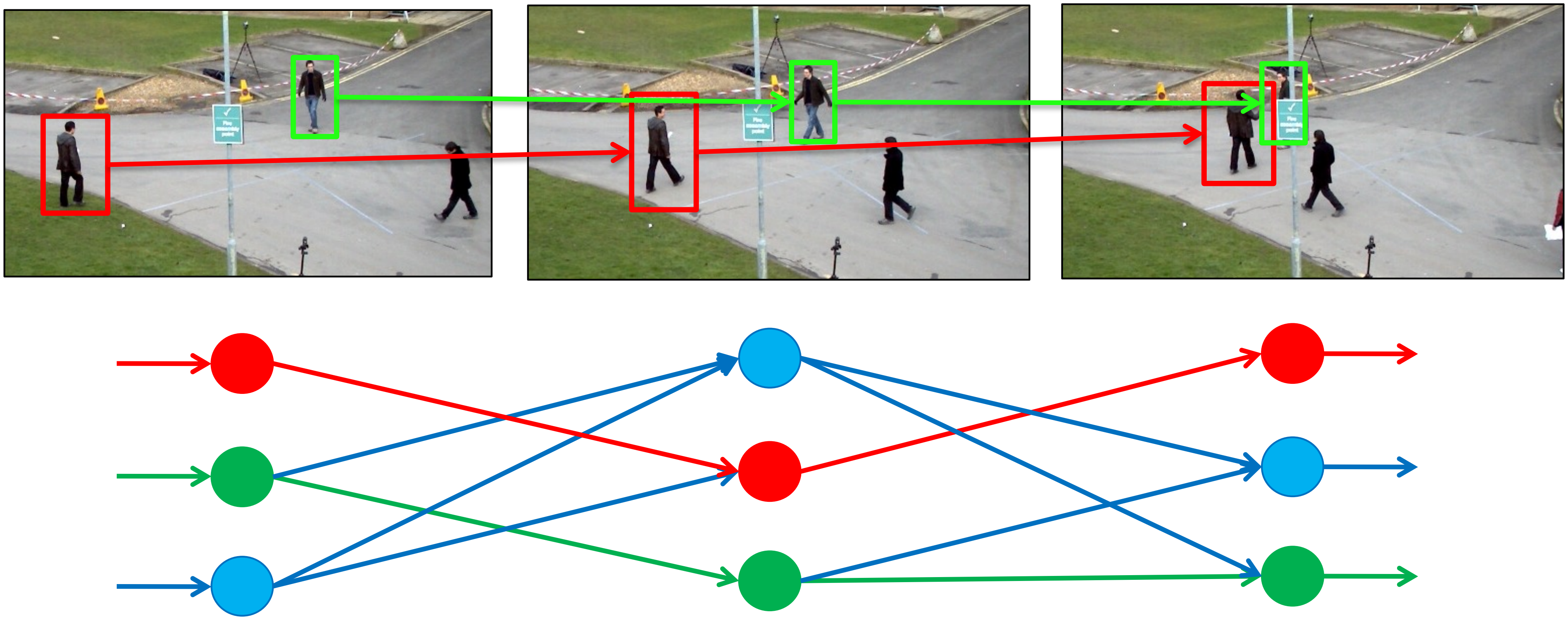
Graphical model

Tracking with network flows



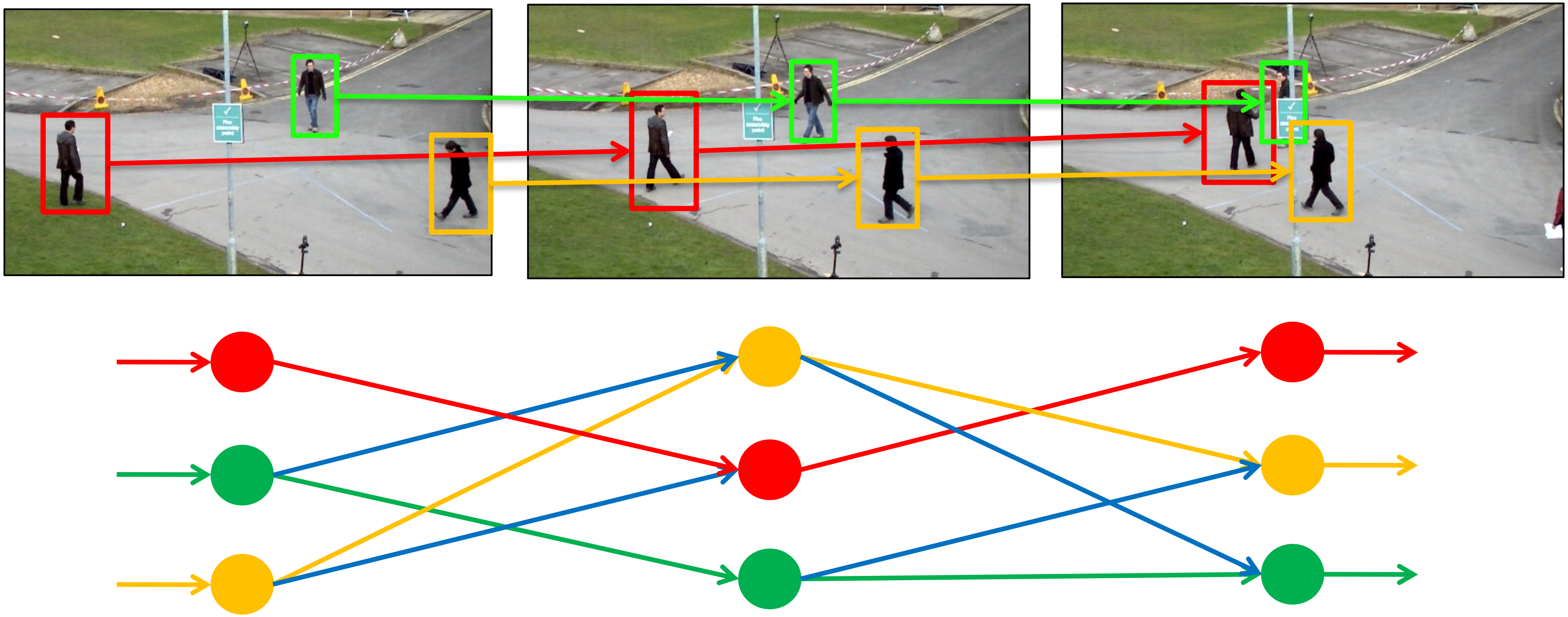
L. Leal-Taixé et al. "Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker." ICCVW2011

Tracking with network flows



L. Leal-Taixé et al. "Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker." ICCVW2011

Tracking with network flows



L. Leal-Taixé et al. "Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker." ICCVW2011

Tracking with network flows

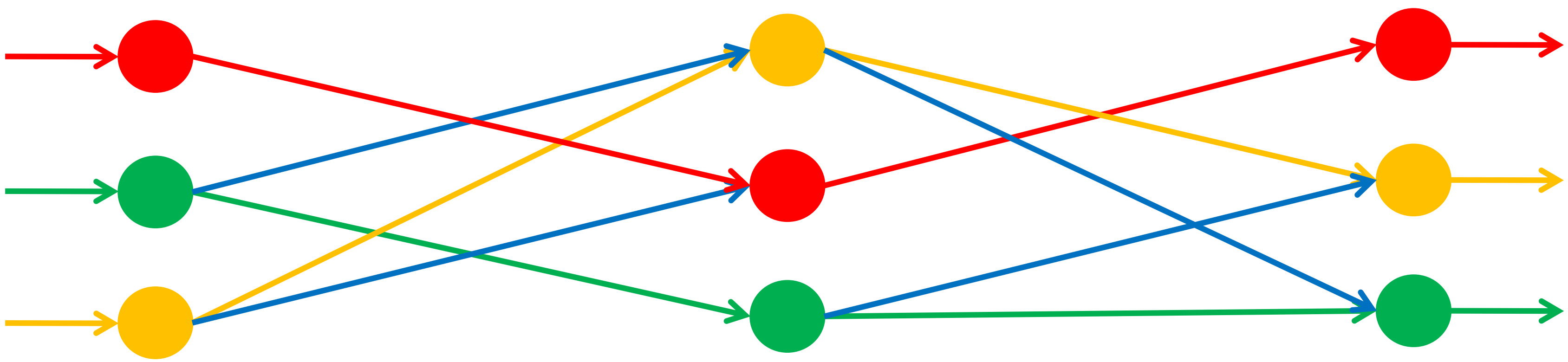
- Node = detection
- Edge = flow = trajectory
- 1 unit of flow = 1 pedestrian

L. Leal-Taixé et al. "Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker." ICCVW2011

Tracking with network flows

- Solving the Minimum Cost Flow Problem

"Determine the minimum cost of shipment of a commodity through a network"

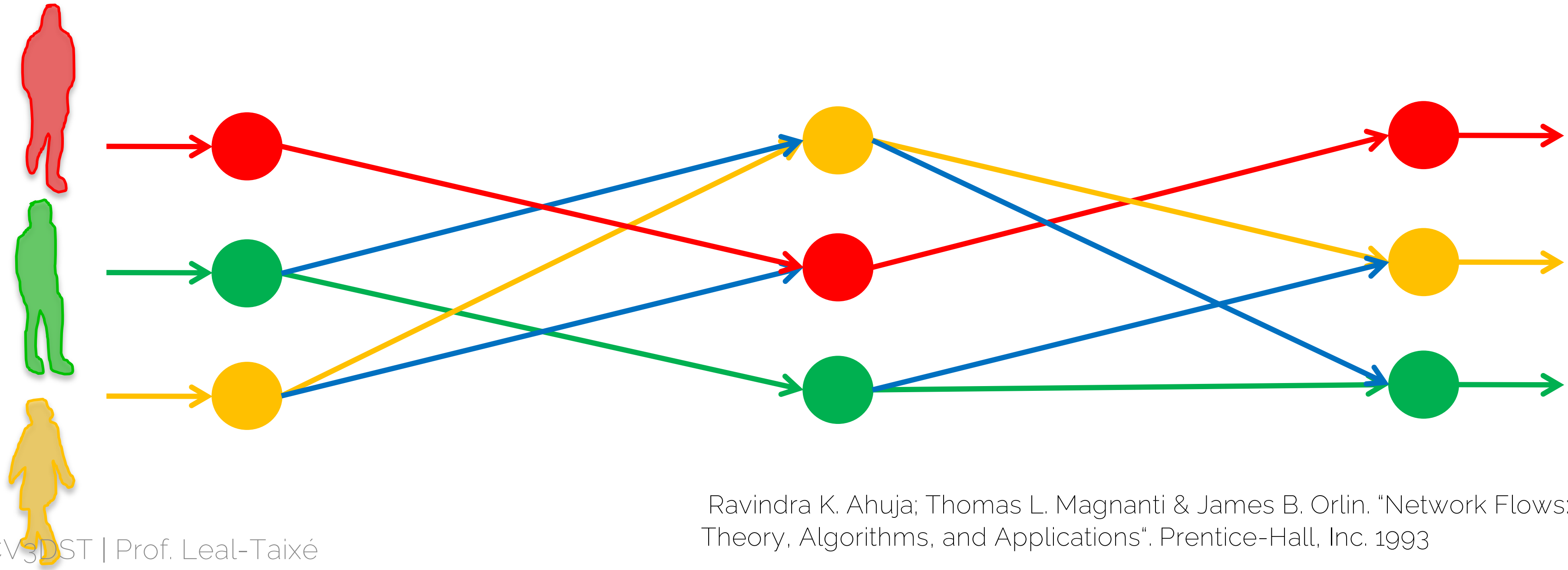


Ravindra K. Ahuja; Thomas L. Magnanti & James B. Orlin. "Network Flows: Theory, Algorithms, and Applications". Prentice-Hall, Inc. 1993

Tracking with network flows

- Solving the Minimum Cost Flow Problem

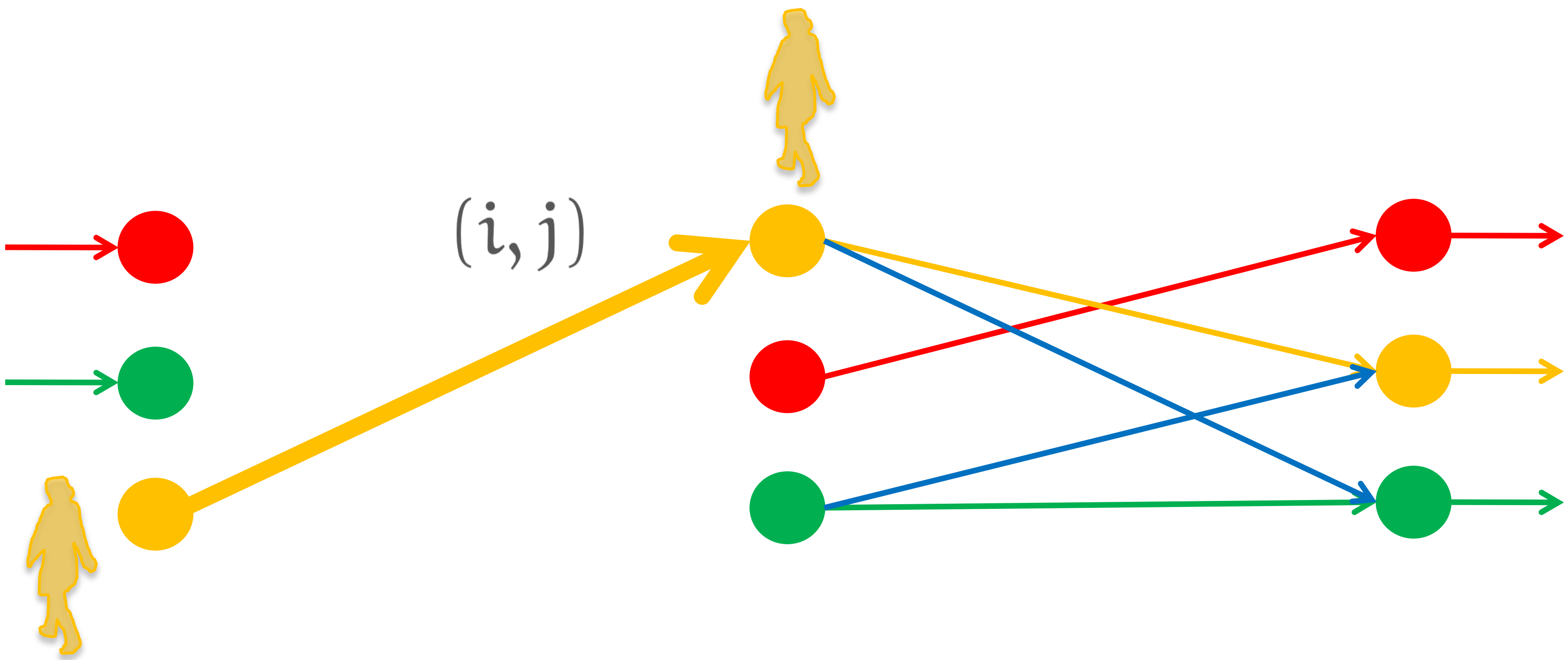
"Determine the minimum cost of shipment of a commodity through a network"



Ravindra K. Ahuja; Thomas L. Magnanti & James B. Orlin. "Network Flows: Theory, Algorithms, and Applications". Prentice-Hall, Inc. 1993

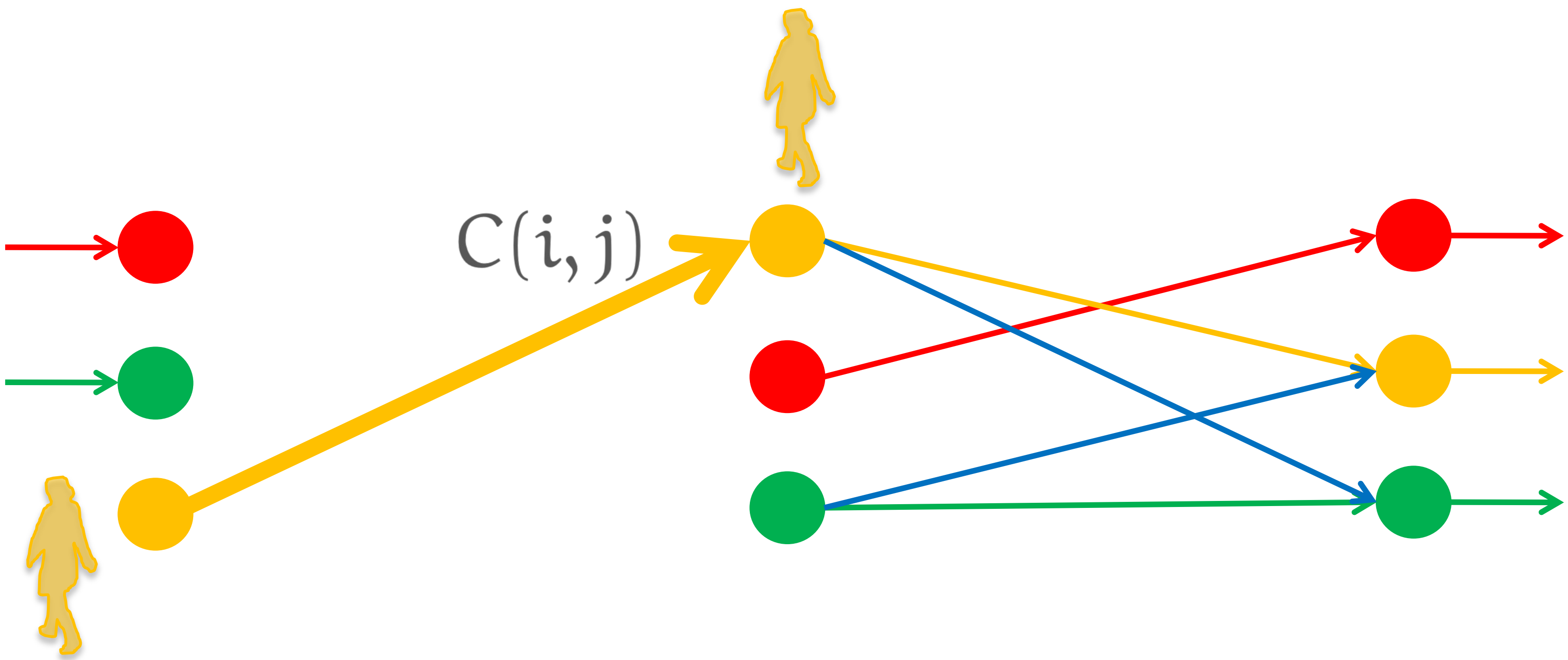
Tracking with network flows

- Objective function $\mathcal{T}* = \arg\min_{\mathcal{T}} \sum_{i,j} C(i,j)f(i,j)$



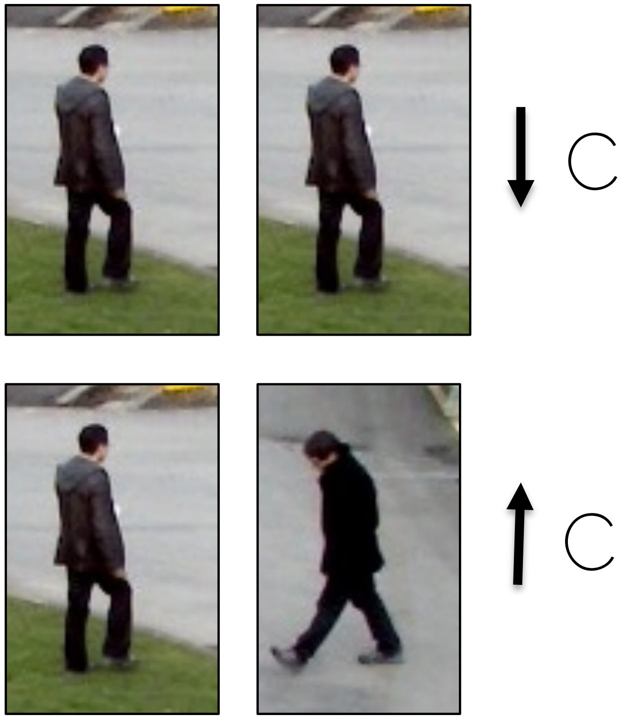
Tracking with network flows

- Objective function $\mathcal{T}* = \arg\min_{\mathcal{T}} \sum_{i,j} C(i,j) f(i,j)$



Tracking with network flows

- Objective function $$\mathcal{T}* = \arg\min_{\mathcal{T}} \sum_{i,j} C(i,j) f(i,j)$$

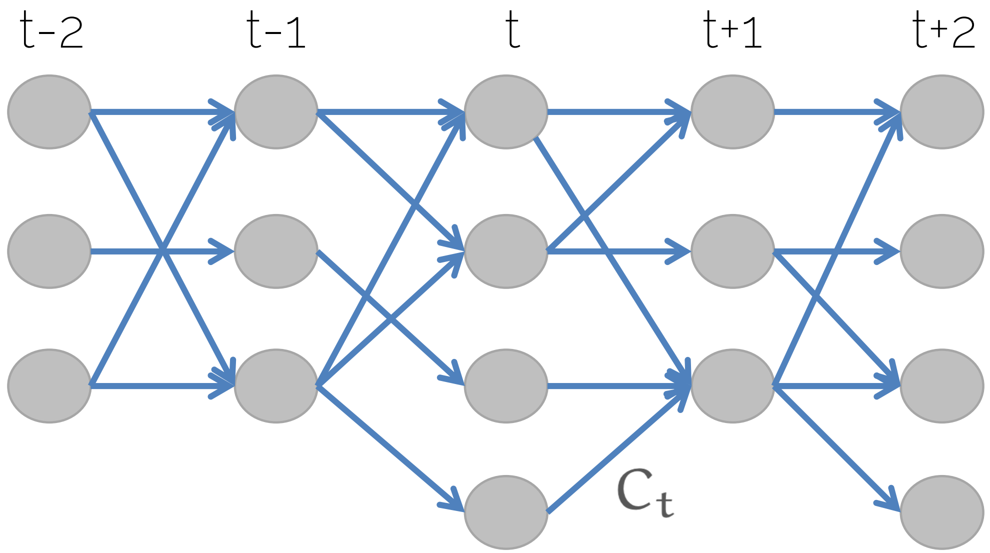


Optimal set of trajectories

Costs – what will drive the tracking

Indicator {0,1}

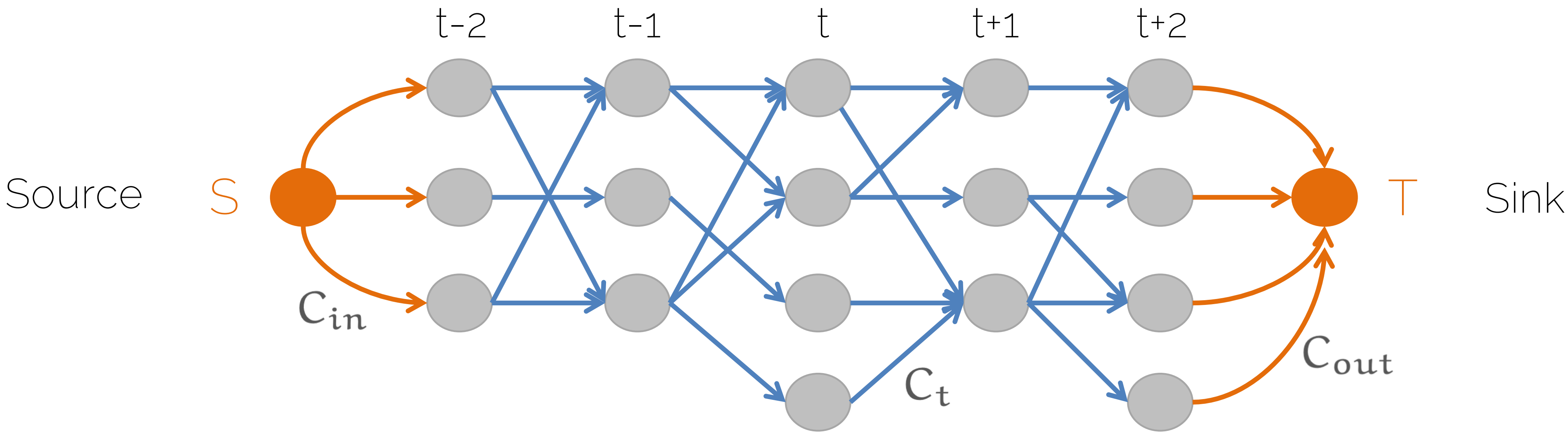
Tracking with network flows



Transition: cost $\propto$ distance between detections

FLOW = TRAJECTORY = PEDESTRIAN

Tracking with network flows

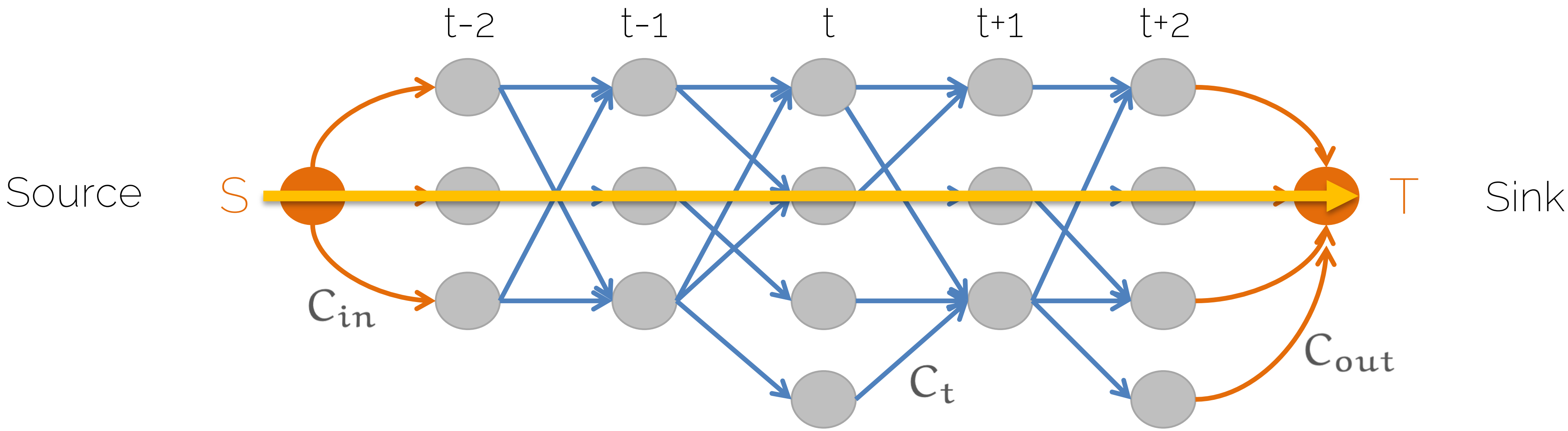


Entrance/exit: cost to start or end a trajectory

Transition: cost $\propto$ distance between detections

FLOW = TRAJECTORY = PEDESTRIAN

Tracking with network flows

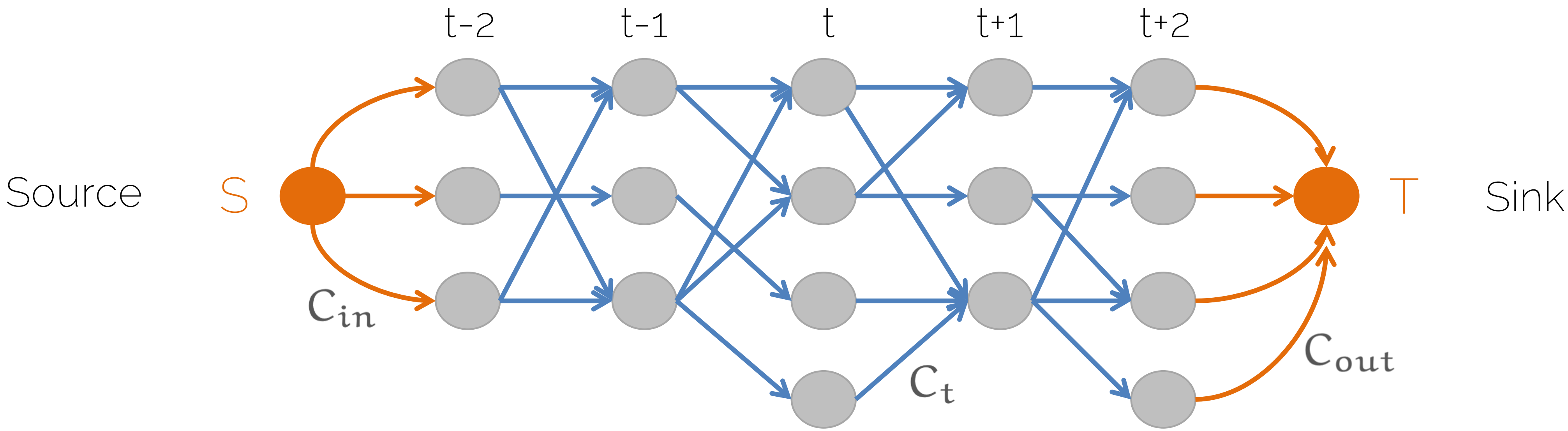


Entrance/exit: cost to start or end a trajectory

Transition: cost $\propto$ distance between detections

Flow can only start at the source node and end at the sink node

Tracking with network flows

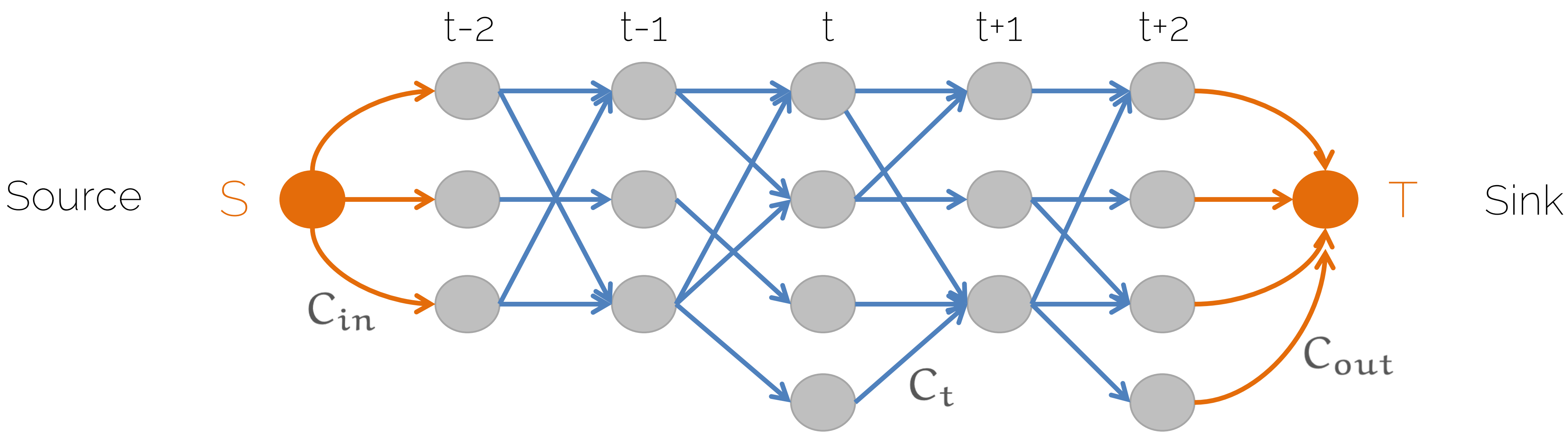


Entrance/exit: cost to start or end a trajectory

Transition: cost $\propto$ distance between detections

What happens if all costs are positive? $\mathcal{T}* = \arg\min_{\mathcal{T}} \sum_{i,j} C(i,j) f(i,j)$

Tracking with network flows



Entrance/exit: cost to start or end a trajectory

Transition: cost $\propto$ distance between detections

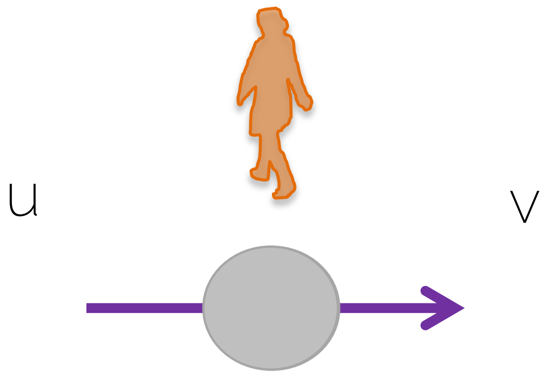
What happens if all costs are positive? $\mathcal{T}* = \arg\min_{\mathcal{T}} \sum_{i,j} C(i,j) f(i,j)$

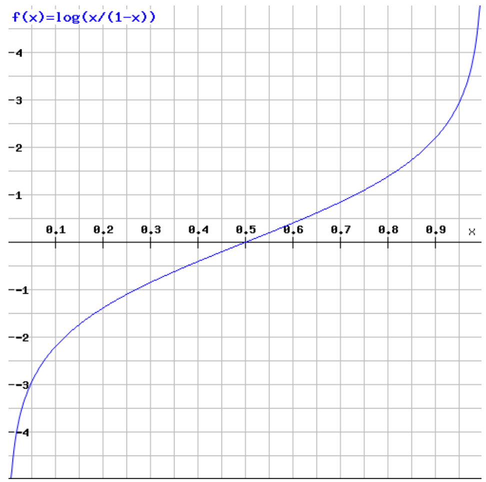
Trivial solution: zero flow!

Tracking with network flows

- We need a negative cost

Detection edge



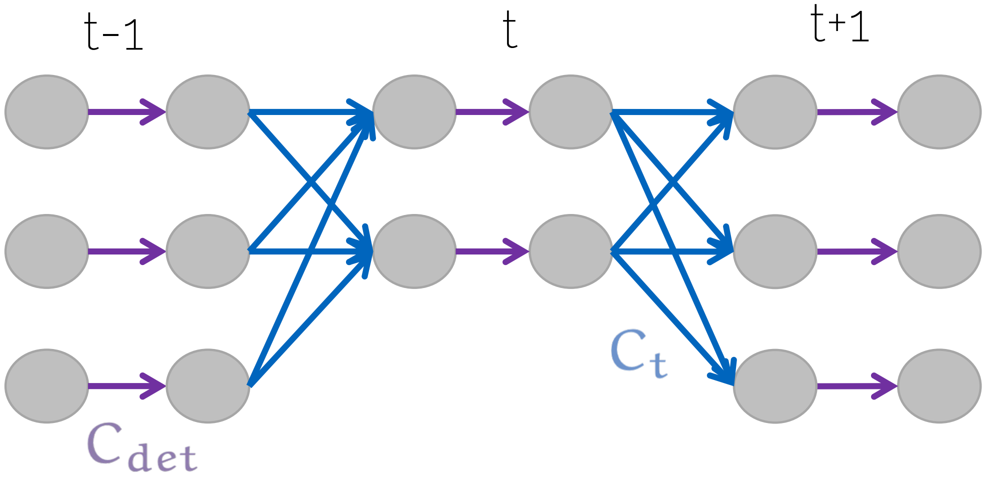


$$C_{det} = \log \frac{\beta_i}{1-\beta_i}$$

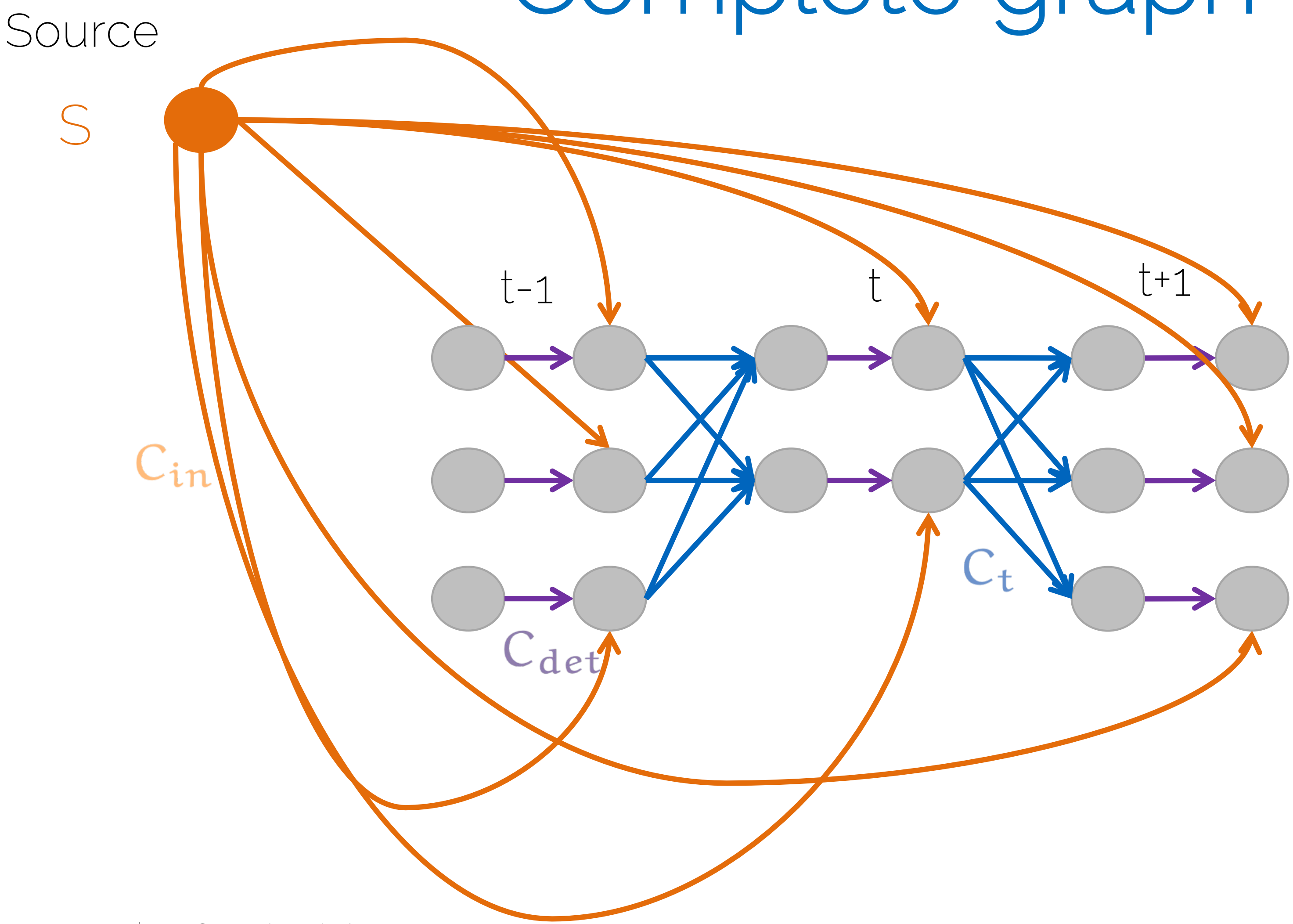
Probability that detection i is a false alarm

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

Complete graph



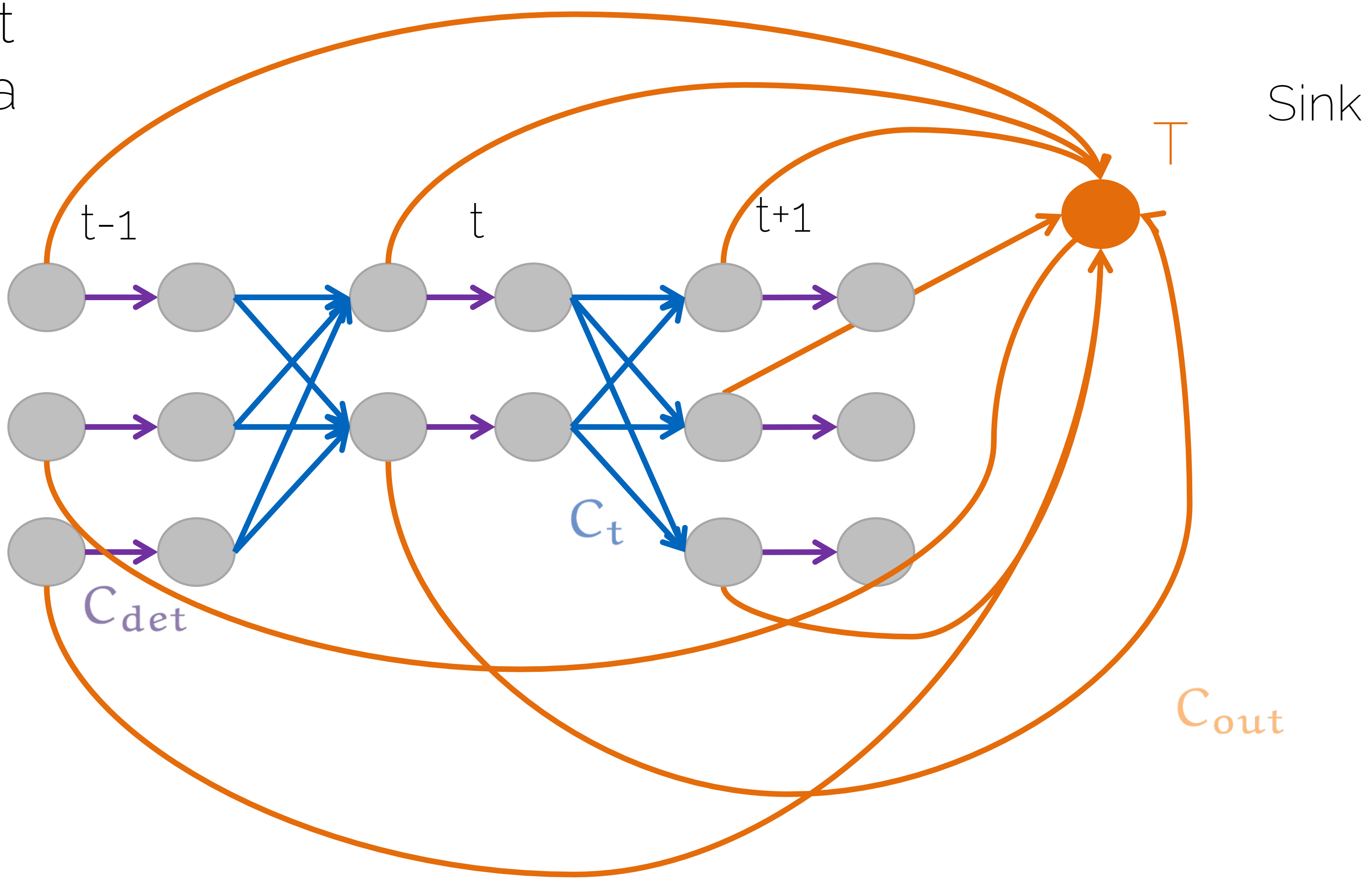
Complete graph



Connections that allow us to start a trajectory

Complete graph

Connections that allow us to end a trajectory



Linear Program MOT formulation

Why a linear program?

- Fast solvers (e.g., Simplex algorithm)

- Guaranteed to converge to the global optimum

Minimum cost flow problem

- Objective function

$$\mathcal{T}* = \arg\min_{\mathcal{T}} \sum_{i,j} C(i,j) f(i,j)$$

Constraints

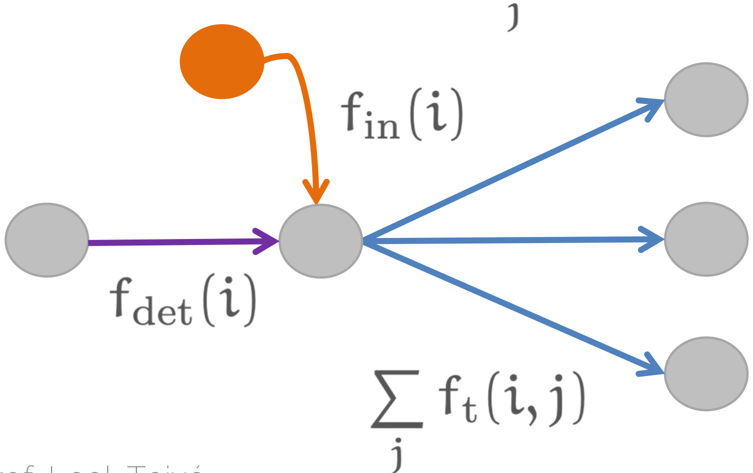
- Objective function

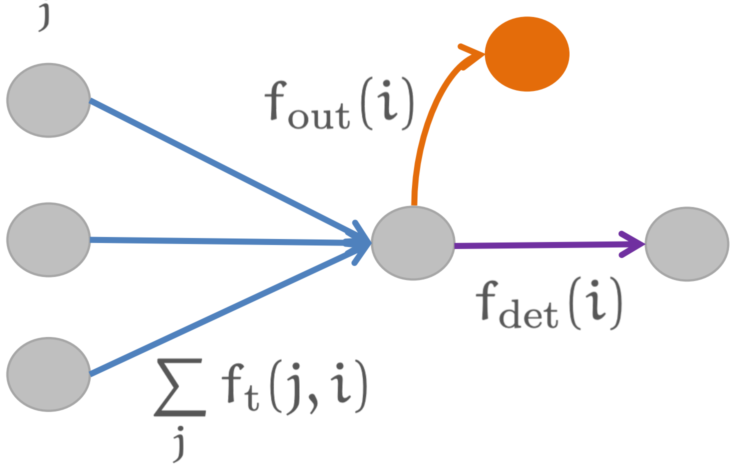
$$\mathcal{T}* = \arg\min_{\mathcal{T}} \sum_{i,j} C(i,j) f(i,j)$$

- Subject to

Flow conservation at the nodes

$$f_{in}(i) + f_{det}(i) = \sum_j f_t(i,j) \qquad \sum_j f_t(j,i) = f_{out}(i) + f_{det}(i)$$





Constraints

- Objective function

$$\mathcal{T}* = \arg\min_{\mathcal{T}} \sum_{i,j} C(i,j) f(i,j)$$

- Subject to

Flow conservation at the nodes

$$f_{in}(i) + f_{det}(i) = \sum_j f_t(i,j) \qquad \sum_j f_t(j,i) = f_{out}(i) + f_{det}(i)$$

Edge capacities

NP-hard!!

$$f_{in}(i) + f_{det}(i) \in \{0,1\} \qquad f_{out}(i) + f_{det}(i) \in \{0,1\} \qquad f \in \{0,1\}$$

LP relaxation

- Objective function

$$\mathcal{T}* = \arg\min_{\mathcal{T}} \sum_{i,j} C(i,j) f(i,j)$$

- Subject to

Flow conservation at the nodes

$$f_{in}(i) + f_{det}(i) = \sum_j f_t(i,j) \qquad \sum_j f_t(j,i) = f_{out}(i) + f_{det}(i)$$

Edge capacities

LP-relaxation

$$0 \leq f_{in}(i) + f_{det}(i) \leq 1 \qquad 0 \leq f_{out}(i) + f_{det}(i) \leq 1 \qquad 0 \leq f \leq 1$$

Solver

- Objective function

$$\mathcal{T}* = \arg\min_{\mathcal{T}} \sum_{\mathbf{i},\mathbf{j}} \mathbf{C}(\mathbf{i},\mathbf{j})\mathbf{f}(\mathbf{i},\mathbf{j})$$

Given the shape of the constraints (total unimodularity), we solve the relaxed problem and still get integer solutions.

Objective function

- Objective function

$$\mathcal{T}* = \arg\min_{\mathcal{T}} \sum_i C_{in}(i) f_{in}(i) + \sum_{i,j} C_t(i,j) f_t(i,j) + \sum_i C_{det}(i) f_{det}(i) + \sum_i C_{out}(i) f_{out}(i)$$

$$\Big\downarrow \; C(i) = -\log P(i)$$

- Equivalent to Maximum a-posteriori tracking formulation

$$\mathcal{T}* = \arg\max_{\mathcal{T}} \prod_j P(\mathbf{o}_j|\mathcal{T}) P(\mathcal{T})$$

Two ways forward

- 1. Improving the costs (aka more learning)
  - 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.

Two ways forward

- 2. Making the graph more complex (including more connections)
  - M. Keuper et al. „Motion segmentation and multiple object tracking by correlation co-clustering". PAMI 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

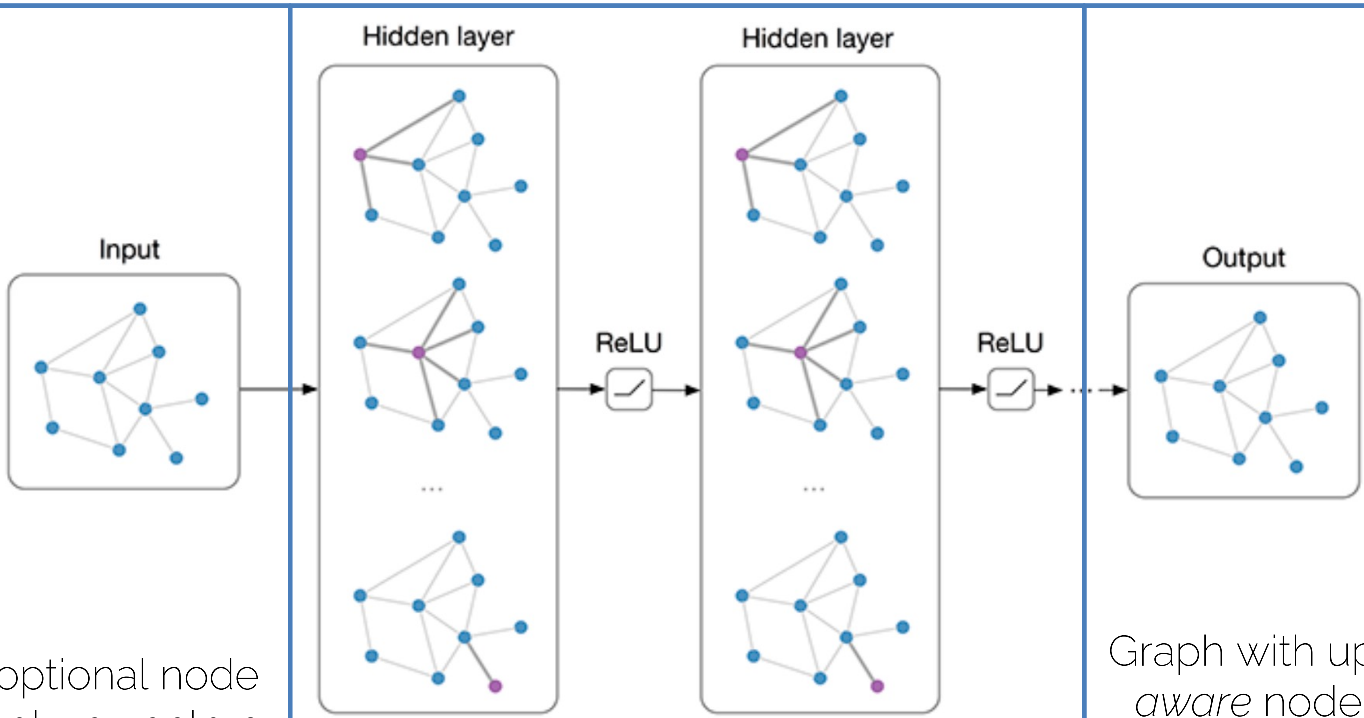
End-to-end learning?

- Can we learn:
  - Features for multi-object tracking (e.g., costs)
  - To do data association, i.e., find a solution on the graph

- We will exploit the graph structure we have just seen and perform end-to-end learning

Message Passing Networks

General Idea



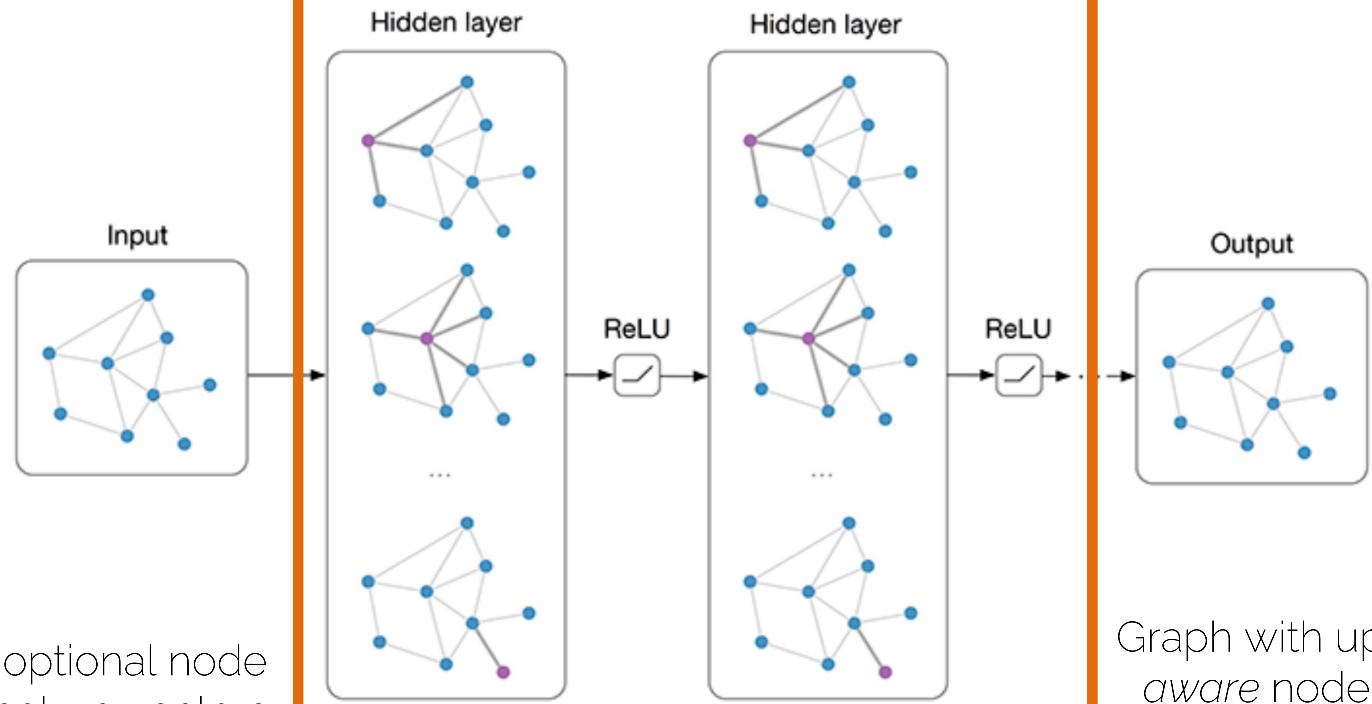
Graph with optional node and edge feature vectors

Information propagation across the graph for several iterations

Graph with updated *context-aware* node and (possibly edge) feature vector(s)

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

General Idea



Graph with optional node and edge feature vectors

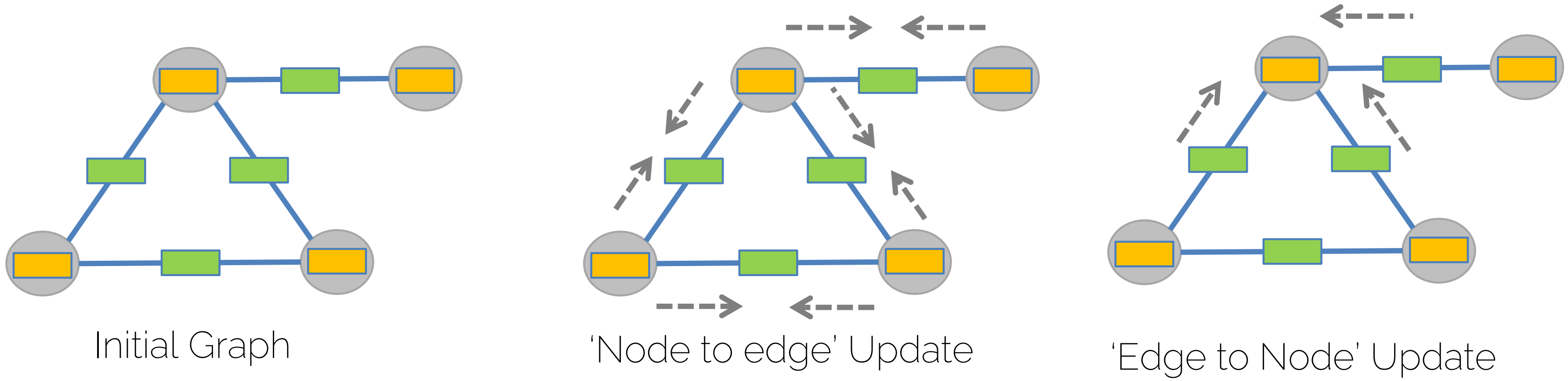
Information propagation across the graph for several iterations

Graph with updated *context-aware* node and (possibly edge) feature vector(s)

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

Learning to propagate information

- We can divide the propagation process in two steps: 'node to edge' and 'edge to node' updates.



Repeat these two updates for a fixed number of iterations (message passing steps) in order to encode context into embeddings

Node embeddings
Edge embeddings

'Node to edge' updates

- Notation:
  - Graph: $G = (V, E)$
  - Initial embeddings: $h^{(0)}_{(i,j)}, (i,j) \in E \quad h^{(0)}_i, i \in V$
  - Embeddings after l steps: $h^{(l)}_{(i,j)}, (i,j) \in E \quad h^{(l)}_i, i \in V$

Embedding of edge that connects nodes I and j

Embedding of node i

'Node to edge' updates

- Notation:
  - Graph: $G = (V, E)$
  - Initial embeddings: $h^{(0)}_{(i,j)}, (i,j) \in E \quad h^{(0)}_i, i \in V$
  - Embeddings after l steps: $h^{(l)}_{(i,j)}, (i,j) \in E \quad h^{(l)}_i, i \in V$
- At every message passing step l , first do:

$$h^{(l)}_{(i,j)} = \mathcal{N}_e \left([h^{(l-1)}_i, h^{(l-1)}_{(i,j)}, h^{(l-1)}_j] \right)$$

Embedding of node i in the precious message passing step

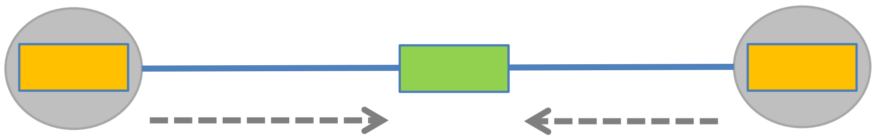
Embedding of edge (i,j) i in the precious message passing step

Embedding of node j in the precious message passing step

'Node to edge' updates

- Notation:
  - Graph: $G = (V, E)$
  - Initial embeddings: $h^{(0)}_{(i,j)}, (i,j) \in E \quad h^{(0)}_i, i \in V$
  - Embeddings after l steps: $h^{(l)}_{(i,j)}, (i,j) \in E \quad h^{(l)}_i, i \in V$
- At every message passing step l, first do:

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

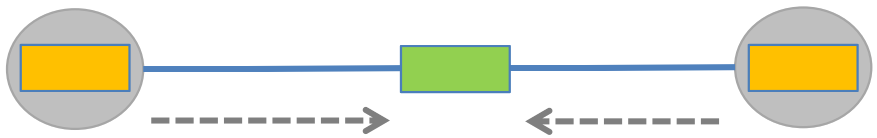


Combine node and edge embeddings

‘Node to edge’ updates

- Notation:
  - Graph: $G = (V, E)$
  - Initial embeddings: $h^{(0)}_{(i,j)}, (i,j) \in E \quad h^{(0)}_i, i \in V$
  - Embeddings after l steps: $h^{(l)}_{(i,j)}, (i,j) \in E \quad h^{(l)}_i, i \in V$
- At every message passing step l, first do:

$$h^{(l)}_{(i,j)} = \underbrace{\mathcal{N}_e}_{\text{Learnable function (e.g. MLP) wtih shared weights across the entire graph}} \left([h_i^{(l-1)}, h_{(i,j)}^{(l-1)}, h_j^{(l-1)}] \right)$$



Combine node and edge embeddings

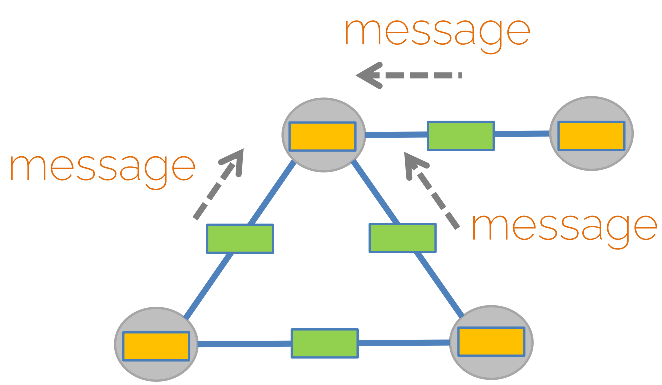
‘Edge to node’ updates

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

'Edge to node' updates

- After a round of edge updates, each edge embedding contains information about its pair of incident nodes
- Then, edge embeddings are used to update nodes:

$$m_{(i,j)}^{(l)} = \underbrace{\mathcal{N}_v}_{\text{Learnable function (e.g. MLP) with shared weights across the entire graph}} \left([h_i^{(l-1)}, h_{(i,j)}^{(l)}] \right)$$



‘Edge to node’ updates

- After a round of edge updates, each edge embedding contains information about its pair of incident nodes
- Then, edge embeddings are used to update nodes:

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

$$h_i^{(l)} = \underbrace{\Phi}_{\substack{\text{Permutation invariant operation} \\ \text{(e.g. sum, mean, max)}}}\left(\left\{m_{(i,j)}^{(l)}\right\}_{j \in \underbrace{N_i}_{\text{Neighbors of node i}}}\right)$$

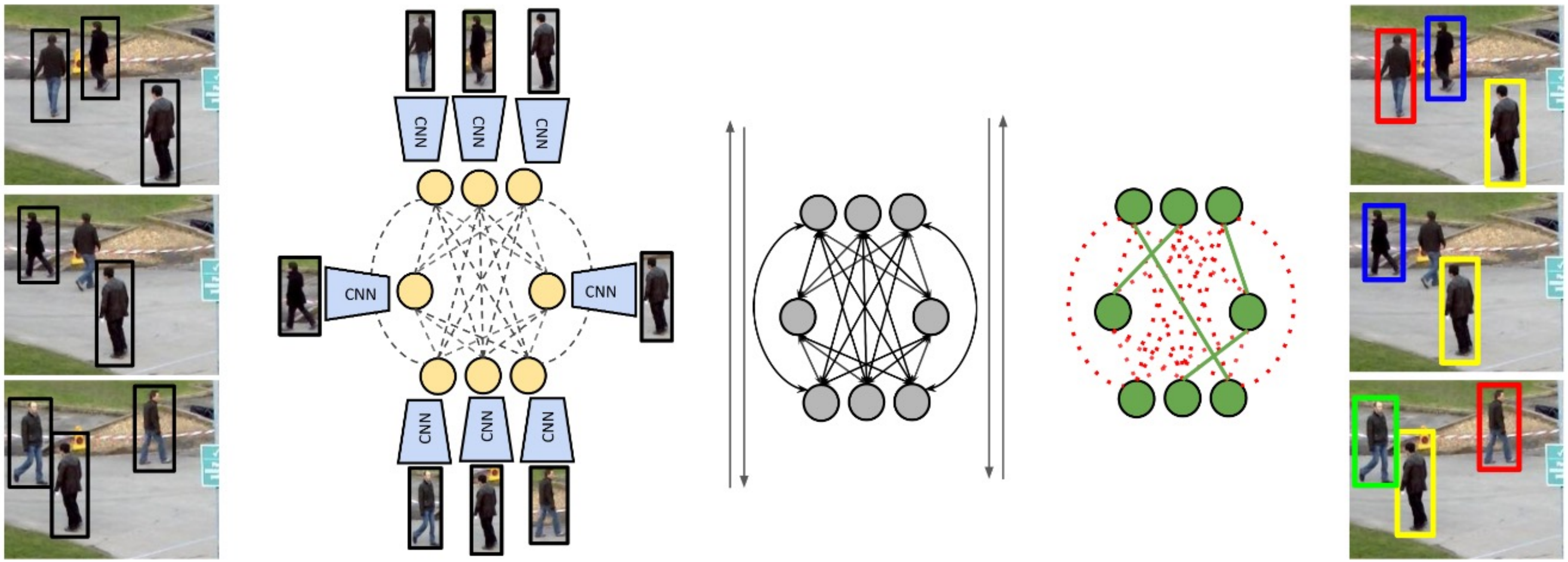
The aggregation provides each node embedding with contextual information about its neighbors

Remarks

- **Main goal**: obtaining node and edge embeddings that contain *context information* encoding graph topology and neighbor's feature information.
- After repeating the node and edge updates for l steps, each node (resp. edge) embedding contains information about all nodes (resp. edge) at distance l (resp. l – 1) → Think of iterations as layers in a CNN
- Observe that all operations used are differentiable, hence, MPNs can be used within end-to-end pipelines
- There is vast literature on different instantiations, as well as variations of the MPN framework we presented. See Battaglia et al. for an extensive review.

MOT with Message Passing Networks

Overview

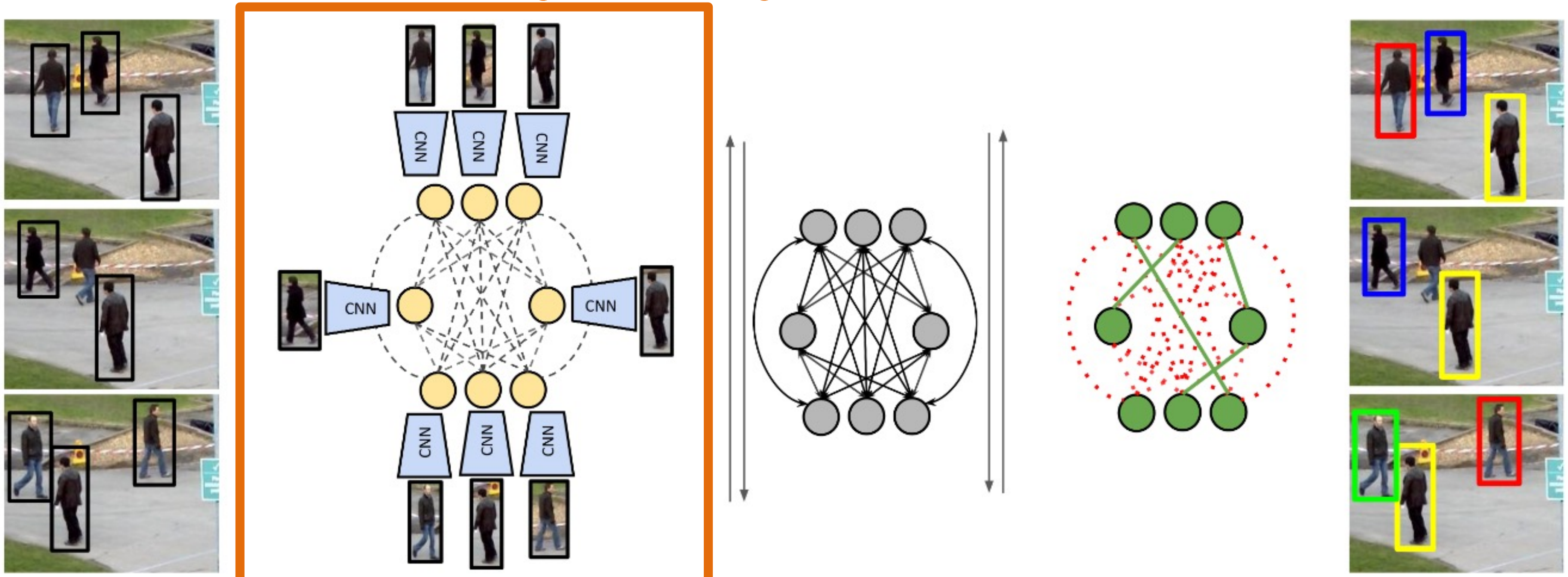


(a) Input (b) Graph Construction + Feature Encoding (c) Neural Message Passing (d) Edge Classification (e) Output

G. Brasó and L. Leal-Taixé. "Learning a Neural Solver for Multiple Object Tracking", arXiv 2019

Overview

Encode appearance and scene geometry cues into node and edge embeddings

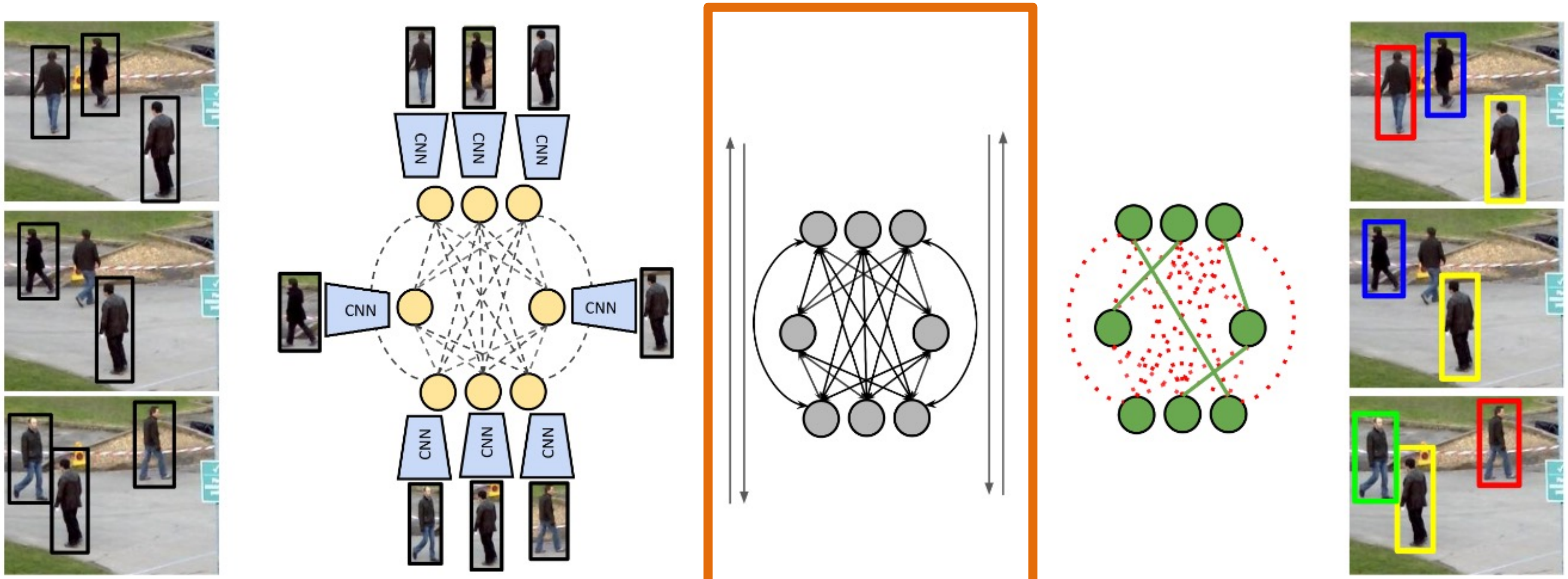


(a) Input (b) Graph Construction + Feature Encoding (c) Neural Message Passing (d) Edge Classification (e) Output

G. Brasó and L. Leal-Taixé. "Learning a Neural Solver for Multiple Object Tracking", arXiv 2019

Overview

Propagate cues across the entire graph with neural message passing

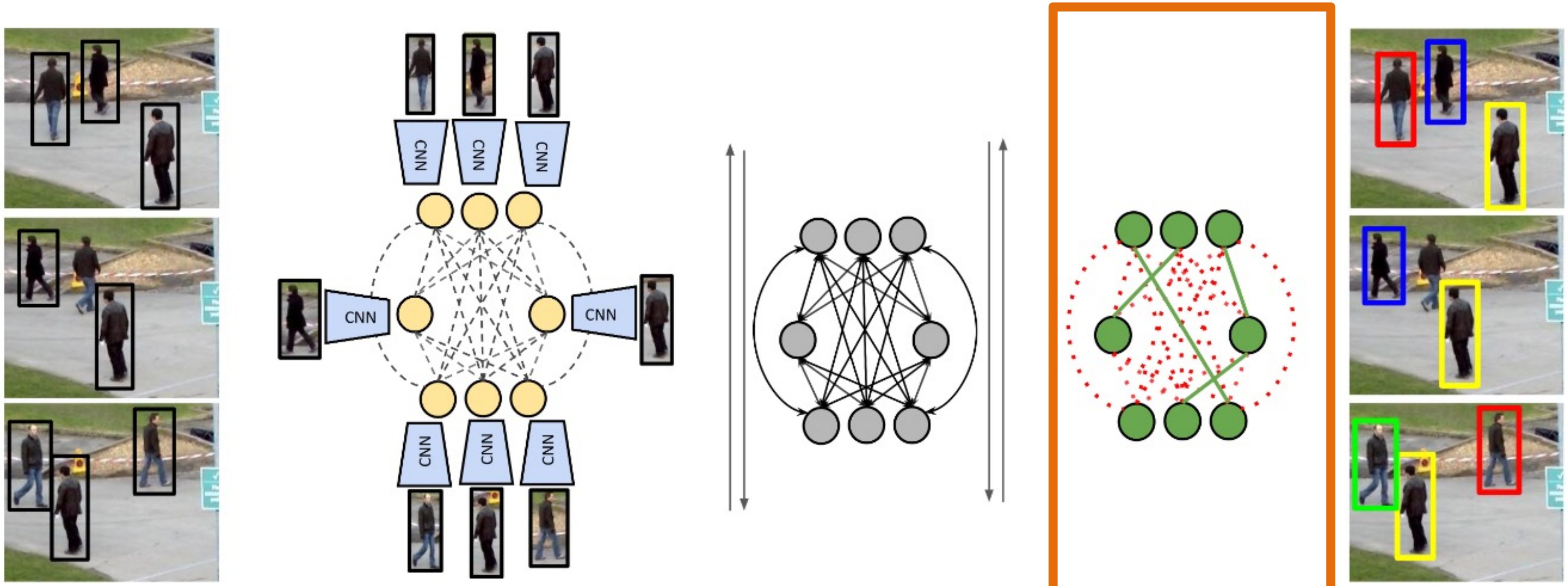


(a) Input (b) Graph Construction + Feature Encoding (c) Neural Message Passing (d) Edge Classification (e) Output

G. Brasó and L. Leal-Taixé. "Learning a Neural Solver for Multiple Object Tracking", arXiv 2019

Overview

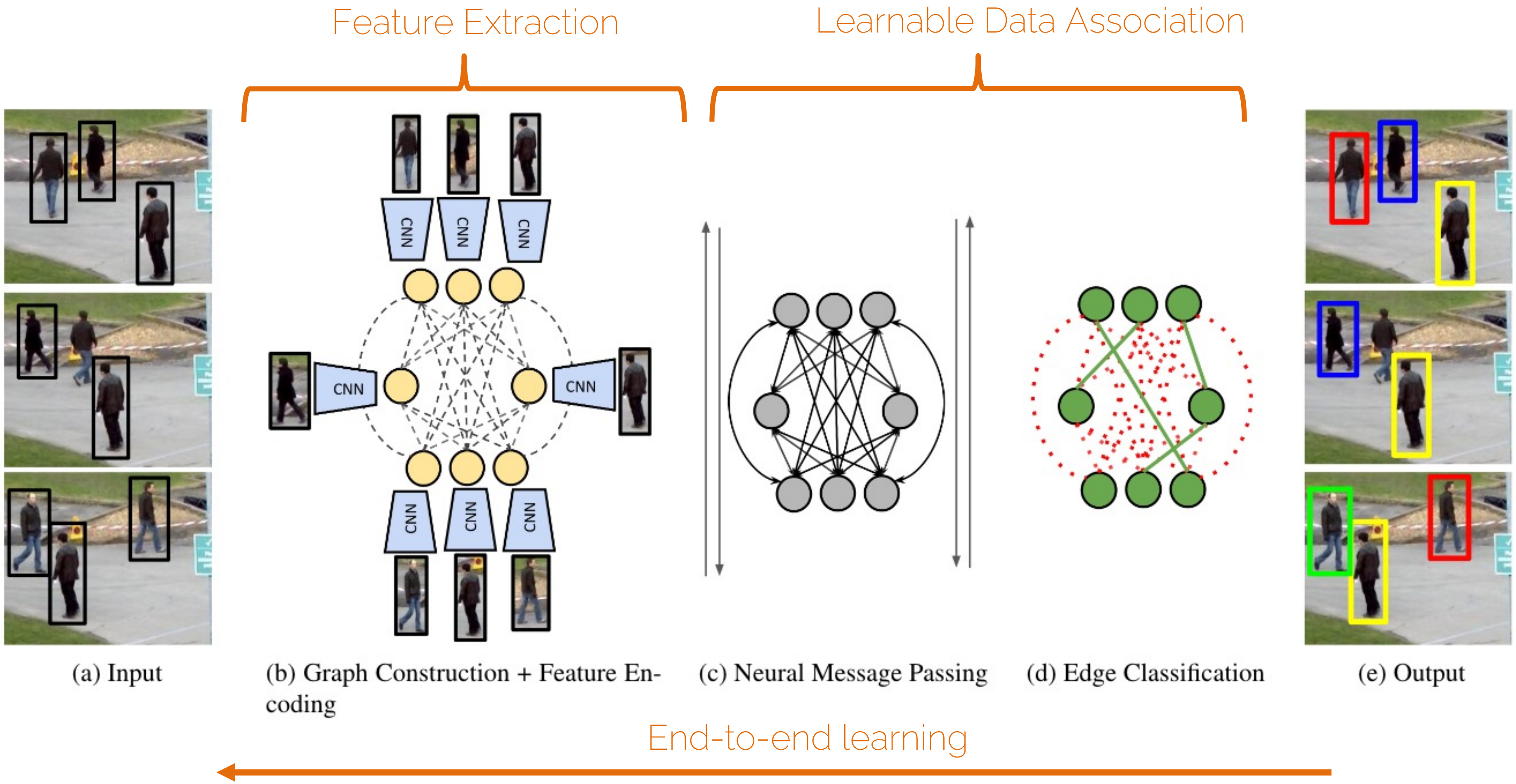
Learn to directly predict solutions of the Min-Cost Flow problem by classifying edge embeddings



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

G. Brasó and L. Leal-Taixé. "Learning a Neural Solver for Multiple Object Tracking", arXiv 2019

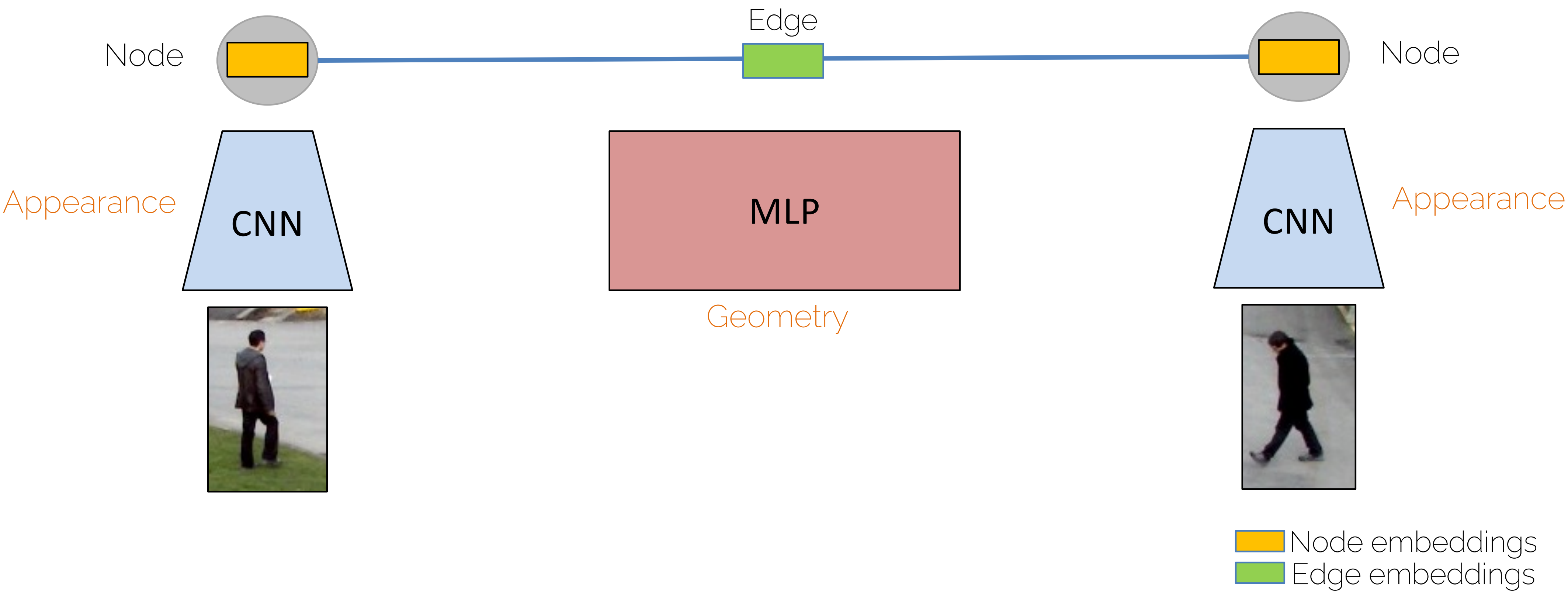
Overview



G. Brasó and L. Leal-Taixé. "Learning a Neural Solver for Multiple Object Tracking", arXiv 2019

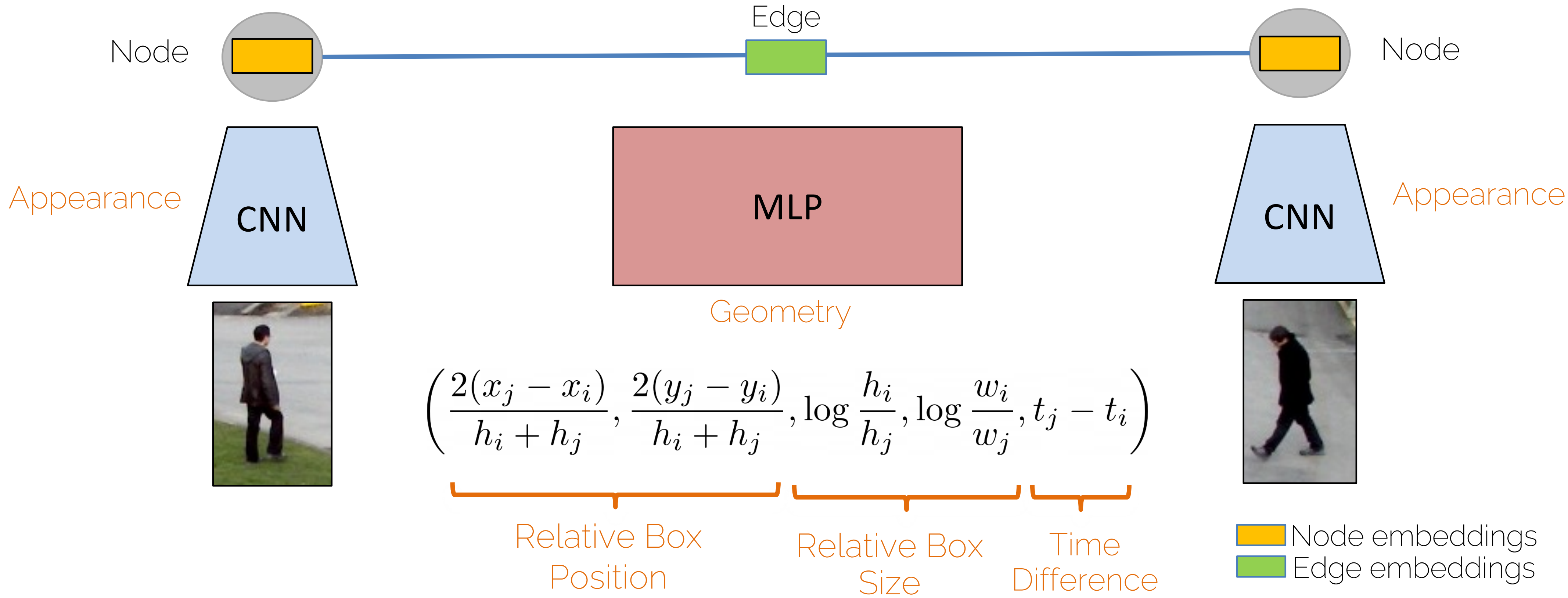
Feature encoding

- Appearance and geometry encodings



Feature encoding

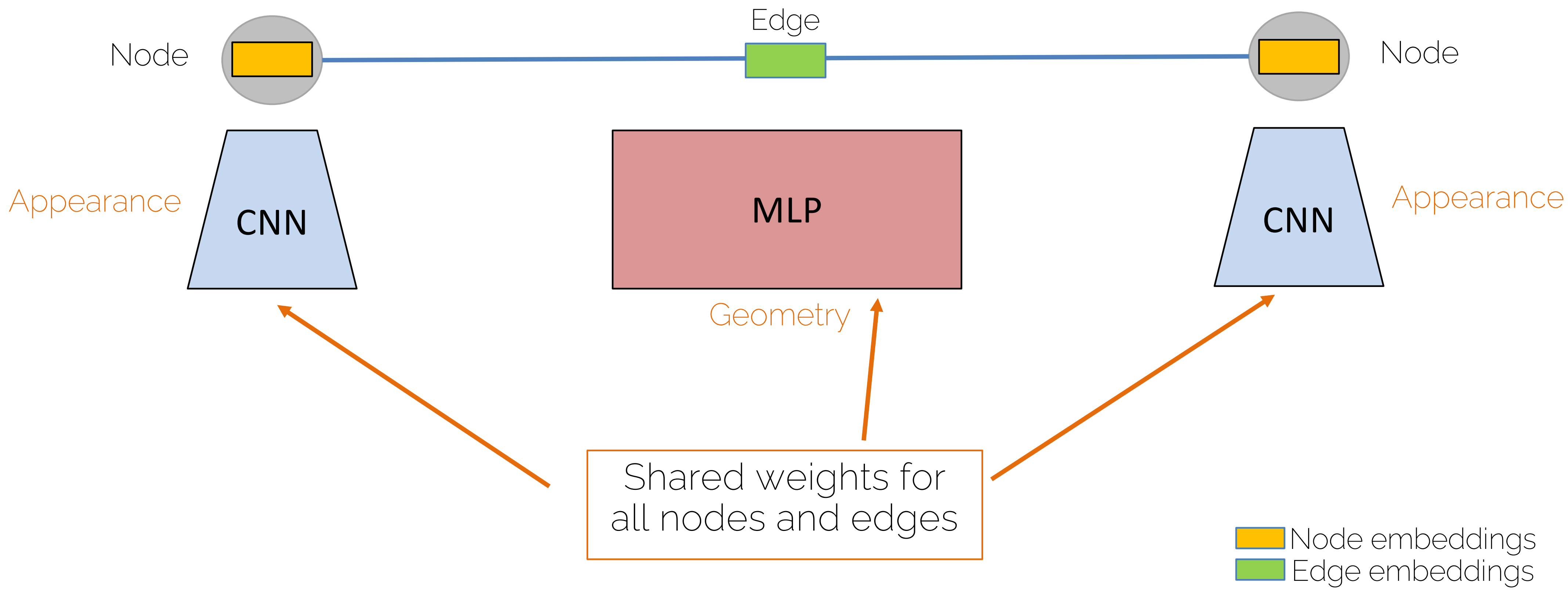
- Appearance and geometry encodings



$$\left(\frac{2(x_j - x_i)}{h_i + h_j}, \frac{2(y_j - y_i)}{h_i + h_j}, \log\frac{h_i}{h_j}, \log\frac{w_i}{w_j}, t_j - t_i\right)$$

Feature encoding

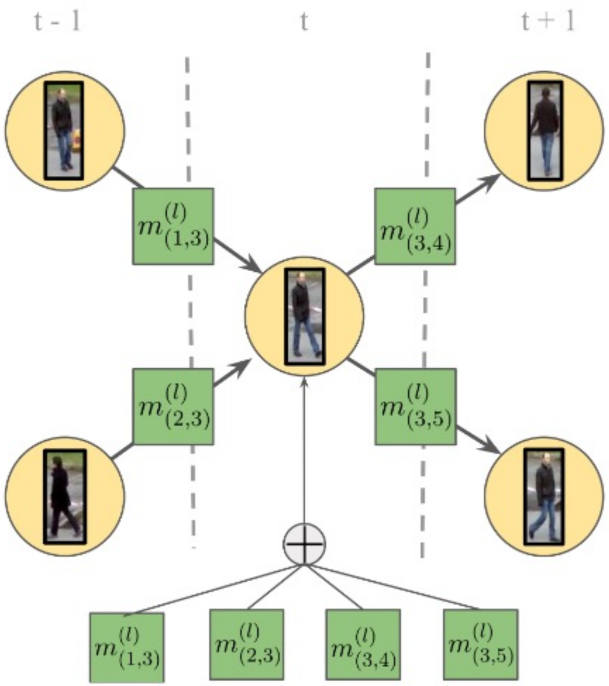
- Appearance and geometry encodings



Feature encoding

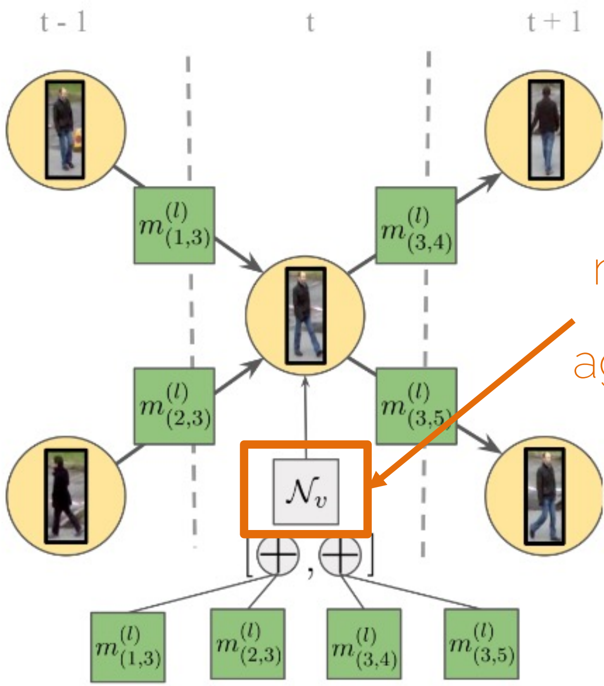
- Instead of defining pairwise costs for edges and unary costs for nodes (classical setting), feature vectors encoding appearance and geometry cues are used
- **Goal:** propagate these embeddings across the entire graph in order to obtain new embeddings encoding high-order information among detections

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

Classifying edges

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

$$\mathcal{L} = \frac{-1}{|E|} \underbrace{\sum_{l=l_0}^{l=L}}_{\text{Sum over the last steps}} \sum_{(i,j)\in E} \underbrace{w}_{\text{Weight to balance active / inactive edges}} \cdot \underbrace{y_{(i,j)} \log(\boxed{\hat{y}^{(l)}_{(i,j)}}) + (1 - y_{(i,j)}) \log(1 - \boxed{\hat{y}^{(l)}_{(i,j)}})}_{\text{Binary cross-entropy}}$$

Edge predictions (w. sigmoid) at iteration l

Some results

- In practice, around 99% 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

G. Brasó and L. Leal-Taixé. "Learning a Neural Solver for Multiple Object Tracking", arXiv 2019

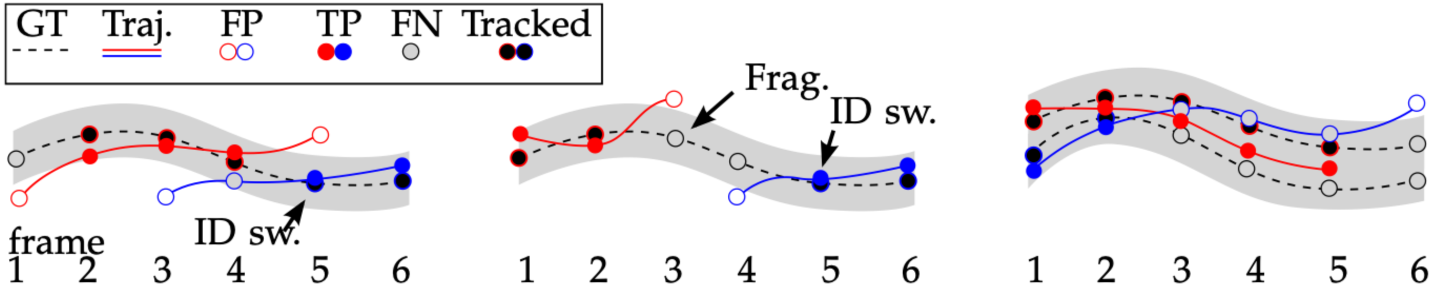
MOT evaluation

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 (missing detections)
  - IDsw: identity switches

Evaluation metrics

- How do we compute ID switches?



(a) (b) (c)

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

(b) You 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. arXiv:1504.01942 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

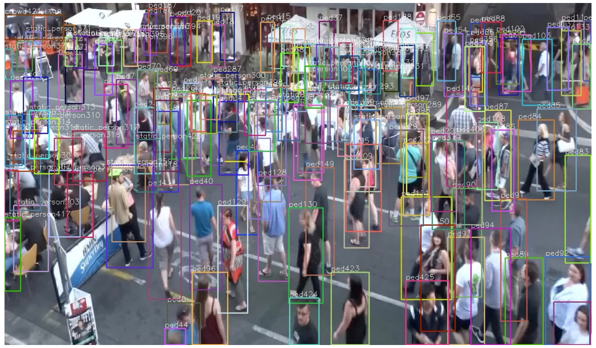
Multi-object tracking accuracy

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

Ground truth

Datasets

- MOTChallenge: www.motchallenge.net (people)
  - Several challenges from less to more crowded

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