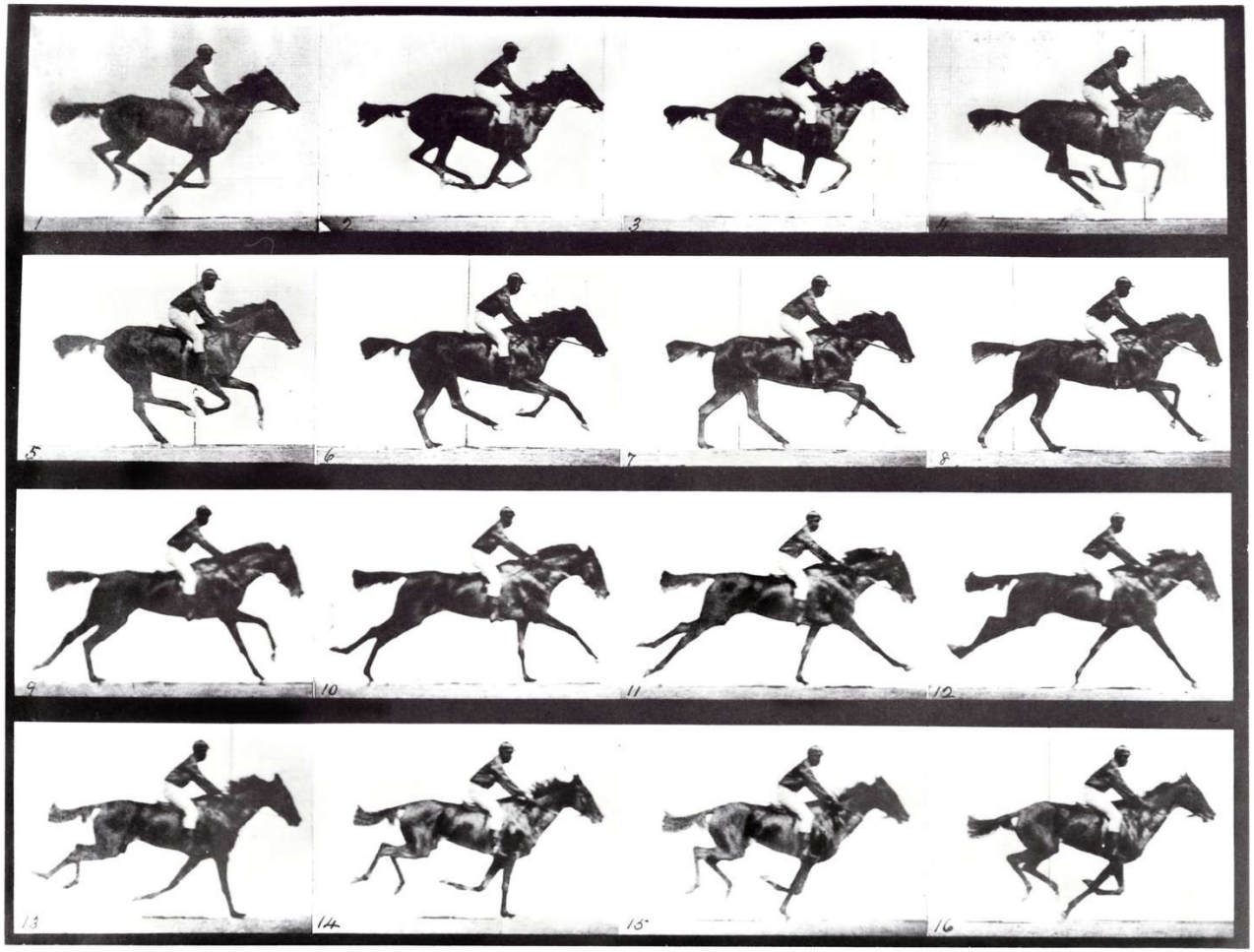
Simple tracking

Marcel Duchamp, Nude
Descending a staircase

Follow an object from frame to frame



Eadweard Muybridge

Tracking – Why?

Motion capture

- build models of moving people from video

Recognition from motion

- eg cyclists move differently than runners

Surveillance

- who is doing what?
  - for security (eg keep people out of sensitive areas in airports)
  - for HCI (eg kinect, eyetoy, etc.)

Tracking - What

Establish state of object using time sequence

- state could be:
  - position; position+velocity; position+velocity+acceleration
  - or more complex, eg all joint angles for a person
- Biggest problem -- Data Association
  - which image pixels are informative, which are not?

Key ideas

- Tracking by detection
  - if we know what an object looks like, that selects the pixels to use
- Tracking through flow
  - if we know how an object moves, that selects the pixels to use

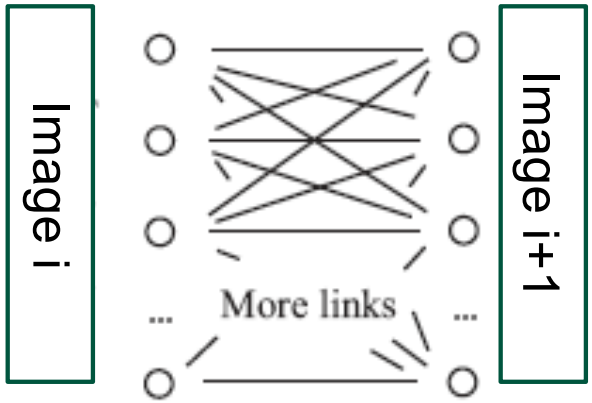
Tracking by detection

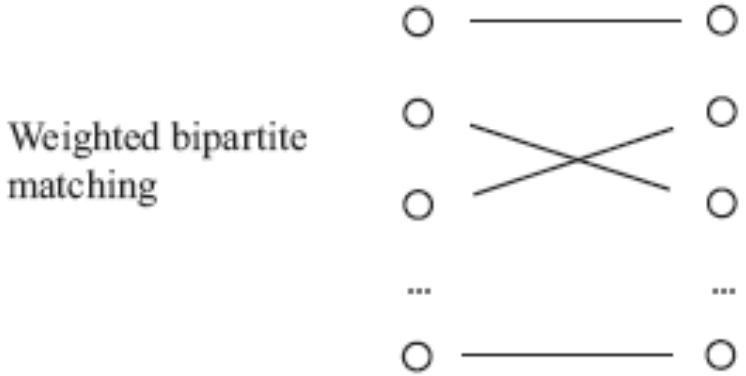
Assume

- a very reliable detector (e.g. faces; back of heads)
- detections that are well spaced in images (or have distinctive properties)
  - e.g. news anchors; heads in public

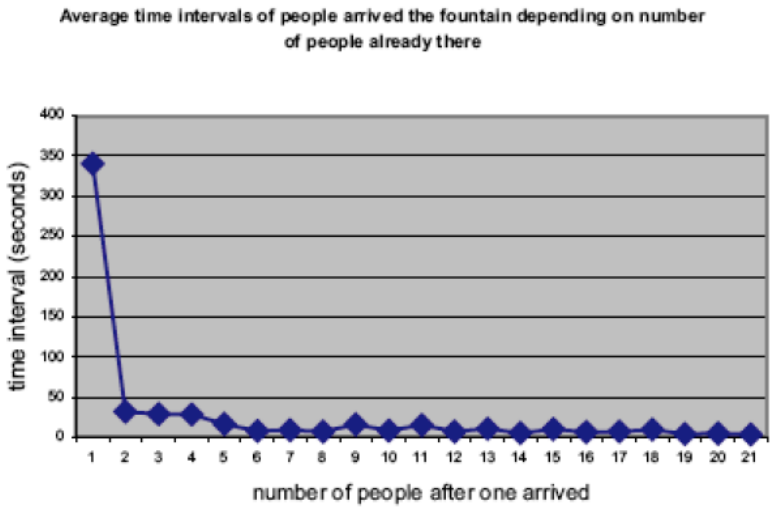
Link detects across time

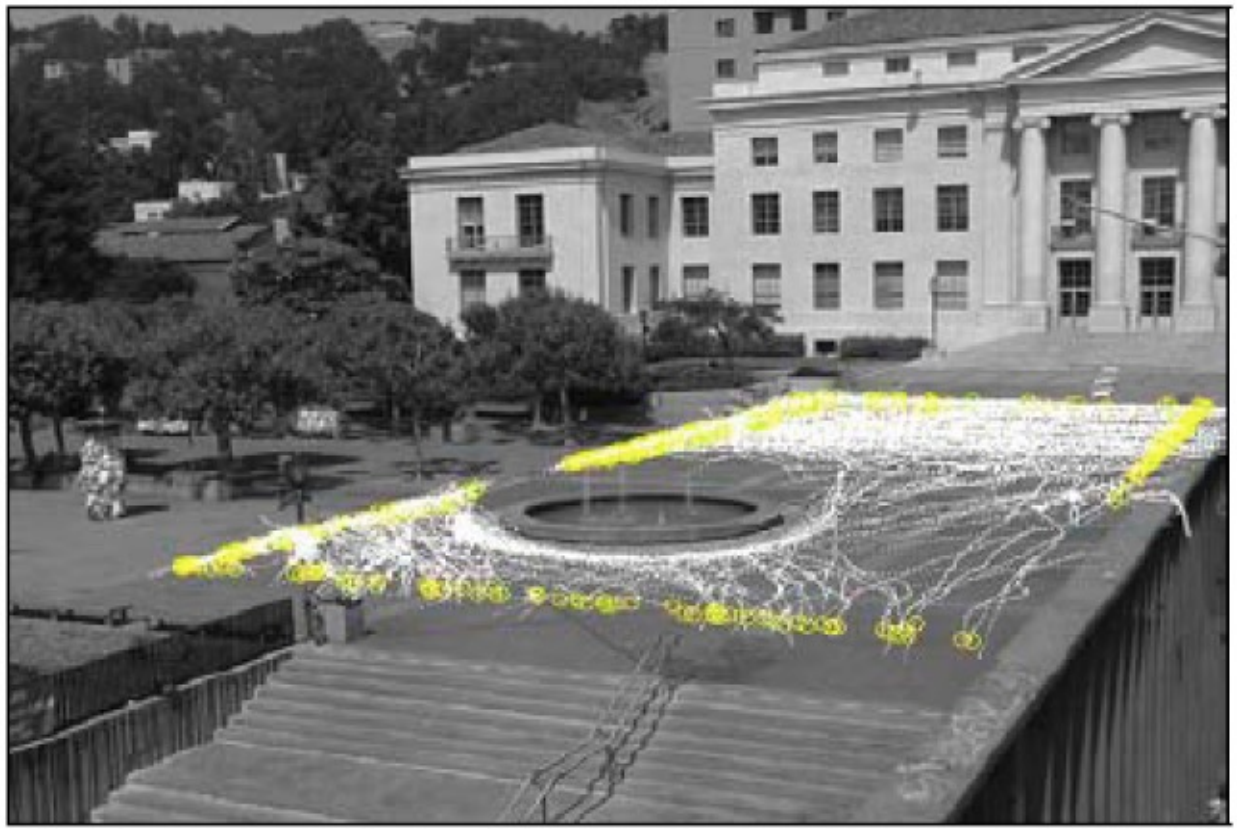
- only one – easy
- More – weighted bipartite matching





Point tracks reveal public behaviour





Yan+Forsyth, 04

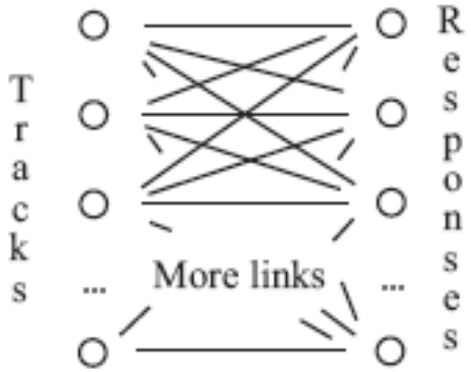
Some detections might fail…

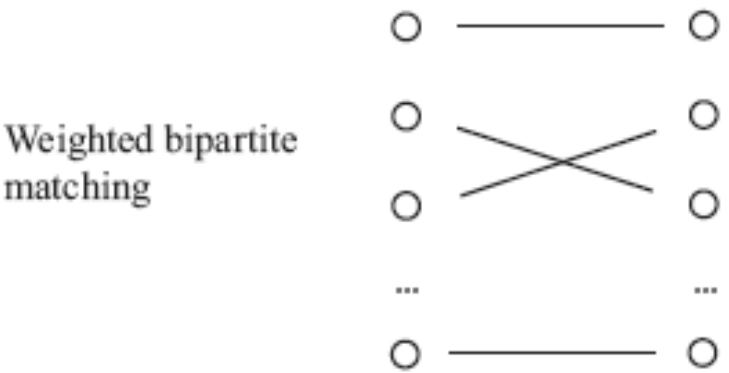
Match measurements to abstract “tracks”

Strategy

- detect in each frame
- link detects to tracks using matching algorithm
  - measurements with no track? create new track
  - tracks with no measurement? wait, then reap
- (perhaps) join tracks over time with global considerations

Link detects to tracks?





Notation:
Write $\mathbf{x}_k(i)$ for the k'th response of the detector in the ith frame
Write $t(k,i)$ for the k'th track in the ith frame
Write $*t(k,i)$ for the detector response attached to the k'th track in the ith frame (Think C pointer notation)

Assumptions: We have a detector which is reasonably reliable.
We know some distance d such that $d(*t(k,i-1), *t(k,i))$ is always small.

First frame: Create a track for each detector response.

N'th frame:
Link tracks and detector responses by solving a bipartite matching problem.
Spawn a new track for each detector response not allocated to a track.
Reap any track that has not received a detector response for some number of frames.

Cleanup: We now have trajectories in space time. Link anywhere this is justified (perhaps by a more sophisticated dynamical or appearance model, derived from the candidates for linking).

Algorithm 11.1: Tracking by Detection.

Issues

What detection and how to detect?

- Interest points
- Detector boxes (but you need to know the category)
- Other kinds of detector report
  - Eg full kinematic body configuration
- Semantic search regions
- User delineated regions
- “Salient” regions

Using both detection and motion information

- Many moving objects move in quite predictable ways
- Use motion and detection together to predict state

Tracking by detection for interest points

Find interest points

Know window (say) in image (n)

Want to find corresponding window in (n+1)

Search over nearby windows

- to find one that minimizes SSD error (Sum of Squared Differences)

$$\sum_{i,j}(\mathcal{R}_{ij}^{(n)} - \mathcal{R}_{ij}^{(n+1)})^2.$$

- where sum is over pixels in rectangle

Matching

Patch is at u, t; moves to u+h, t+1; h is small

Error is sum of squared differences

$$E(\boldsymbol{h}) = \sum_{\boldsymbol{u} \in \mathcal{P}_t} [I(\boldsymbol{u}, t) - I(\boldsymbol{u} + \boldsymbol{h}, t + 1)]^2$$

This is minimized when

$$\nabla_{\boldsymbol{h}} E(\boldsymbol{h}) = 0.$$

substitute

$$I(\boldsymbol{u} + \boldsymbol{h}, t + 1) \approx I(\boldsymbol{u}, t) + \boldsymbol{h}^T \nabla I$$

get

$$\left[\sum_{\boldsymbol{u} \in \mathcal{P}_t} (\nabla I)(\nabla I)^T \right] \boldsymbol{h} = \sum_{\boldsymbol{u} \in \mathcal{P}_t} [I(\boldsymbol{u}, t) - I(\boldsymbol{u}, t + 1)] \nabla I$$

Matching, II

We can tell if the match is good by looking at

$$\left[\sum_{\boldsymbol{u}\in\mathcal{P}_t}(\nabla I)(\nabla I)^T\right]$$

- which will be poorly conditioned if matching is poor
  - eg featureless region
  - eg flow region

Matching, III

Previous test compares i and i+1

But the patch should be “well behaved” over long time scales

Compare N’th frame with first by

Compute affine transform M, c that minimizes LSE

$$E(\mathcal{M}, \boldsymbol{c}) = \sum_{\boldsymbol{u} \in \mathcal{P}_1} \left[I(\boldsymbol{u}, 1) - I(\mathcal{M}\boldsymbol{u} + \boldsymbol{c}, t) \right]^2 .$$

Check value of LSE; too big? Reject track

Efros et al, 03

Interest points yield tracker and matcher

Combine:

Collect multiple frames of person A

Track person B in video

For each box, match to best person A frame

Blend that in

Efros et al, 03

Track by flow (simple form)

Assume

- appearance unknown (but domain, parametric flow model known)
- optic flow assumptions, as before

Initialize

- mark out domain

Track

- choose flow model parameters that align domain in pic n with n+1 best
- push domain through flow model to get new domain

$$\sum_{\mathbf{x}\in\mathcal{D}_t}\left[\mathcal{I}(\mathbf{x},t)-\mathcal{I}(\mathbf{x}+\mathbf{V}(\mathbf{x},\theta),t)\right]^2$$

Issues

What detection and how to detect?

- ~~Interest points~~
- Detector boxes (but you need to know the category)
- Other kinds of detector report
  - Eg full kinematic body configuration
- Semantic search regions
- User delineated regions
- "Salient" regions

These work like Interest points

Issues: how do you describe a region? Region deforms

Using both detection and motion information

- Many moving objects move in quite predictable ways
- Use motion and detection together to predict state

Using a motion model

Many objects move quite predictably

- Movement is slow wrt frame rate, OR
- Acceleration is small, OR
- Acceleration is fixed (eg movement under gravity)

So we know a lot about where the object is in the next frame

this should help our estimate

Imagine detector is noisy, and movement is predictable

A 1-D problem

Drop a measuring device on a cable down a hole

- where is it?

Setup:

- measurement of depth x
- actual distance down the hole θ
- known $p(\theta)$ which will be normal, $N(\theta_c; \sigma_c^2)$
- known $p(x|\theta)$ which will be normal, $N(c\theta; \sigma_m^2)$

Q: what is $p(\theta|x)$?

A 1D problem, II

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)}$$ (Bayes rule), so that:

$$p(\theta|x) \propto p(x|\theta)p(\theta)$$

And:

$$\log p(\theta|x) = \log p(x|\theta) + \log p(\theta) + K$$

$$= -\frac{(c\theta - x)^2}{2\sigma_m^2} - \frac{(\theta - \theta_c)^2}{2\sigma_c^2} + K'$$

A 1D problem, III

$$\log p(\theta|x) = -\frac{(c\theta - x)^2}{2\sigma_m^2} - \frac{(\theta - \theta_c)^2}{2\sigma_c^2} + K'$$

$$= -\frac{\theta^2}{2}\left[\frac{\sigma_m^2 + c^2\sigma_c^2}{\sigma_m^2\sigma_c^2}\right] + \theta\left[\frac{\theta_c\sigma_m^2 + cx\sigma_c^2}{\sigma_m^2\sigma_c^2}\right] + K''$$

Notice that this is a normal distribution! (check that the log is quadratic in theta)

A 1D problem, IV

Now we can recover parameters for $p(\theta|x)$

Write $N(\mu_t; \sigma_t^2)$

$$\log p(\theta|x) = -\frac{(\theta - \mu_t)^2}{2\sigma_t^2} + K'''$$

$$= -\frac{\theta^2}{2\sigma_t^2} + \theta\frac{\mu_t}{\sigma_t^2} + K'''$$

Pattern matching

Compare

$$\log p(\theta|x) = -\frac{(c\theta - x)^2}{2\sigma_m^2} - \frac{(\theta - \theta_c)^2}{2\sigma_c^2} + K'$$

$$= -\frac{\theta^2}{2}\left[\frac{\sigma_m^2 + c^2\sigma_c^2}{\sigma_m^2\sigma_c^2}\right] + \theta\left[\frac{\theta_c\sigma_m^2 + cx\sigma_c^2}{\sigma_m^2\sigma_c^2}\right] + K''$$

and

$$\log p(\theta|x) = -\frac{(\theta - \mu_t)^2}{2\sigma_t^2} + K'''$$

$$= -\frac{\theta^2}{2\sigma_t^2} + \theta\frac{\mu_t}{\sigma_t^2} + K'''$$

A 1D problem, V

$$\mu_t = \frac{\theta_c \sigma_m^2 + cx\sigma_c^2}{\sigma_m^2 + c^2\sigma_c^2}$$

$$\sigma_t^2 = \frac{\sigma_c^2 \sigma_m^2}{\sigma_m^2 + c^2\sigma_c^2}$$

- Important checks
  - What happens when the measurement is unreliable?
  - What happens when the prior is very diffuse?

Summary, with change of notation

Useful Fact: 9.2 *The parameters of a normal posterior with a single measurement*

Assume we wish to estimate a parameter θ. The prior distribution for θ is normal, with known mean μ_π and known standard deviation σ_π. We receive a single data item x_1 and a scale c_1. The likelihood of x_1 is normal with mean $c_1\theta$ and standard deviation $\sigma_{m,1}$, where $\sigma_{m,1}$ is known. Then the posterior, $p(\theta|x_1, c_1, \sigma_{m,1}, \mu_\pi, \sigma_\pi)$, is normal, with mean

$$\mu_1 = \frac{c_1 x_1 \sigma_\pi^2 + \mu_\pi \sigma_{m,1}^2}{\sigma_{m,1}^2 + c_1^2 \sigma_\pi^2}$$

and standard deviation

$$\sigma_1 = \sqrt{\frac{\sigma_{m,1}^2 \sigma_\pi^2}{\sigma_{m,1}^2 + c^2 \sigma_\pi^2}}.$$

Now a second measurement arrives…

We know that $p(\theta|x)$ is normal

- think of this as the prior

We know that $p(x_1|\theta)$ is normal

- think of this as the likelihood

So:

- the posterior $p(\theta|x_1, x)$ must be normal
- and we can update as before!

Key points:

Can represent posteriors easily, cause they're normal

Updates are easy

True in higher dimensions, too, and very Important. We'll cover in some detail.

The 1D Example is too simple…

Notice we had multiple measurements, but the object didn't move

Typical object:

- Start state then iterate
  - Move
  - Generate measurement

Typical tracking/filtering:

- Start with prior
  - Predict new state resulting from movement
  - Update prediction using new measurement

Filtering

The moving object has a state

- Position
- Position and velocity
- Position, velocity, acceleration
- Etc.

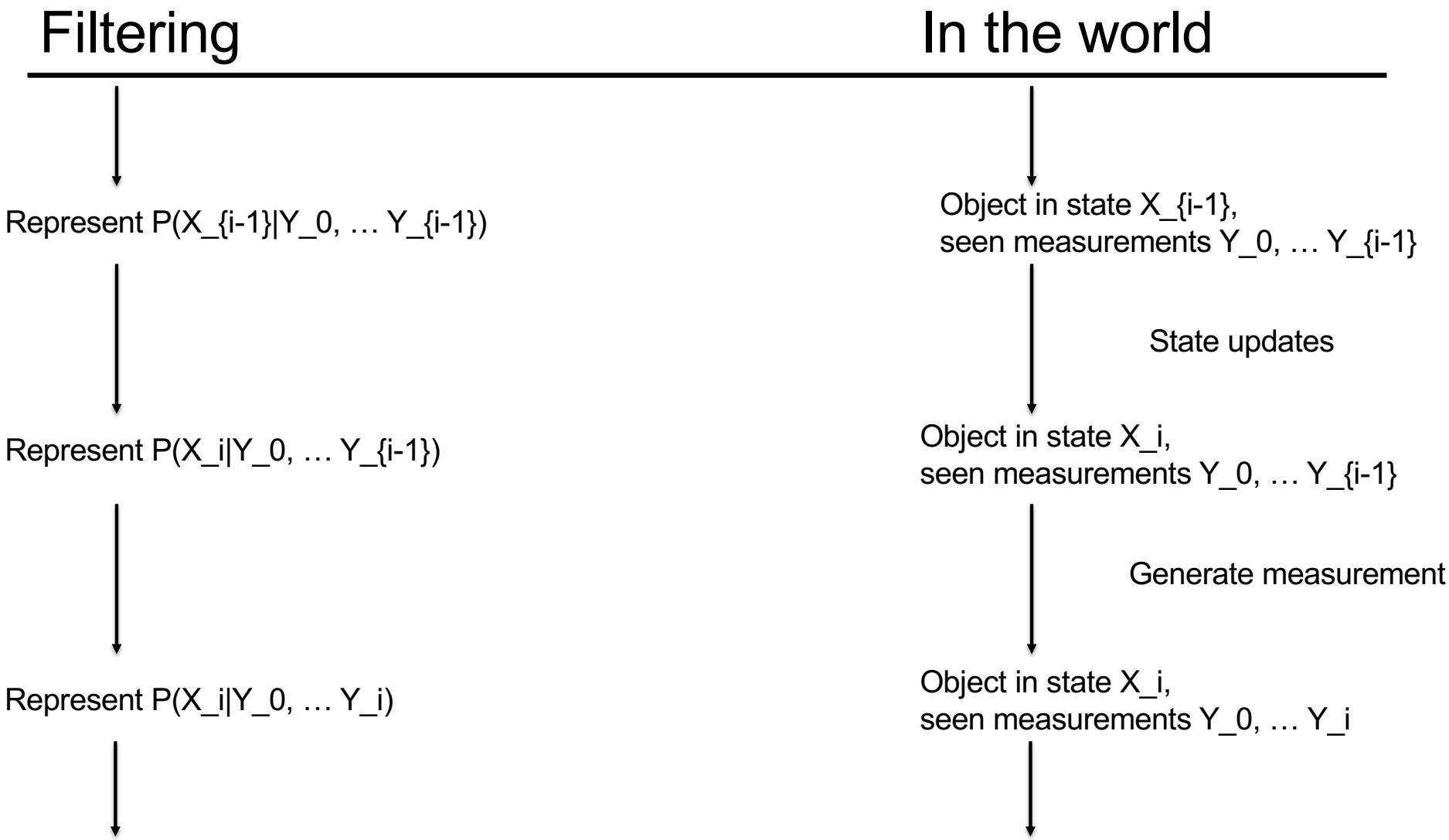
Every tick of time (eg at each frame)

- State is updated by a known (but possibly noisy) procedure
- Equivalently, the object moves
- There is a measurement of state, possibly noisy

Filtering

- Update some representation of state, using measurement and motion information

The big engine: the Kalman Filter


Filtering
In the world
Represent P(X_{i-1}|Y_0, … Y_{i-1})
Represent P(X_i|Y_0, … Y_{i-1})
Represent P(X_i|Y_0, … Y_i)
Object in state X_{i-1},
seen measurements Y_0, … Y_{i-1}
State updates
Object in state X_i,
seen measurements Y_0, … Y_{i-1}
Generate measurement
Object in state X_i,
seen measurements Y_0, … Y_i

Assumption: Linear dynamics and measurement

- State changes as:

Square matrix of full rank

$$\mathbf{x}_i = \mathcal{D}_i \mathbf{x}_{i-1} + \xi$$

This is a normal random variable with zero mean and known covariance

- Measurements are:

Any matrix whose dimensions are OK

$$\mathbf{y}_i = \mathcal{M}_i \mathbf{x}_i + \zeta$$

This is a (different!) normal random variable with zero mean and known covariance

Examples

- Dynamical models
  - Drifting points
    - new state = old state + gaussian noise
  - Points moving with constant velocity
    - new position=old position + (dt) old velocity + gaussian noise
    - new velocity= old velocity+gaussian noise
  - Points moving with constant acceleration
    - etc
- Measurement models
  - state=position; measurement=position+gaussian noise
  - state=position and velocity; measurement=position+gaussian noise
    - but we could infer velocity
  - state=position and velocity and acceleration; measurement=position+gaussian noise

Other notation

Read this as: x_i is normally distributed. The mean is a linear function of x_i-1 and whose variance is known (and can depend on i).

$$\mathbf{x}_i \sim N(\mathcal{D}_i \mathbf{x}_{i-1}; \Sigma_{di})$$

$$\mathbf{y}_i \sim N(\mathcal{M}_i \mathbf{x}_i; \Sigma_{mi})$$

Read this as: y_i is normally distributed. The mean is a linear function of x_i and whose variance is known (and can depend on i)

Important facts

$P(\mathbf{y}_i|\mathbf{x}_i)$ is normal, by construction

Assume that $P(\mathbf{x}_{i-1}|\mathbf{y}_0,\ldots,\mathbf{y}_{i-1})$ is normal

Then $P(\mathbf{x}_i|\mathbf{y}_0,\ldots,\mathbf{y}_{i-1})$ and $P(\mathbf{x}_i|\mathbf{y}_0,\ldots,\mathbf{y}_i)$ are normal

But this means that if P(X_0) is normal, all distributions are normal (induction!)

Important, because normal distributions are easy to represent

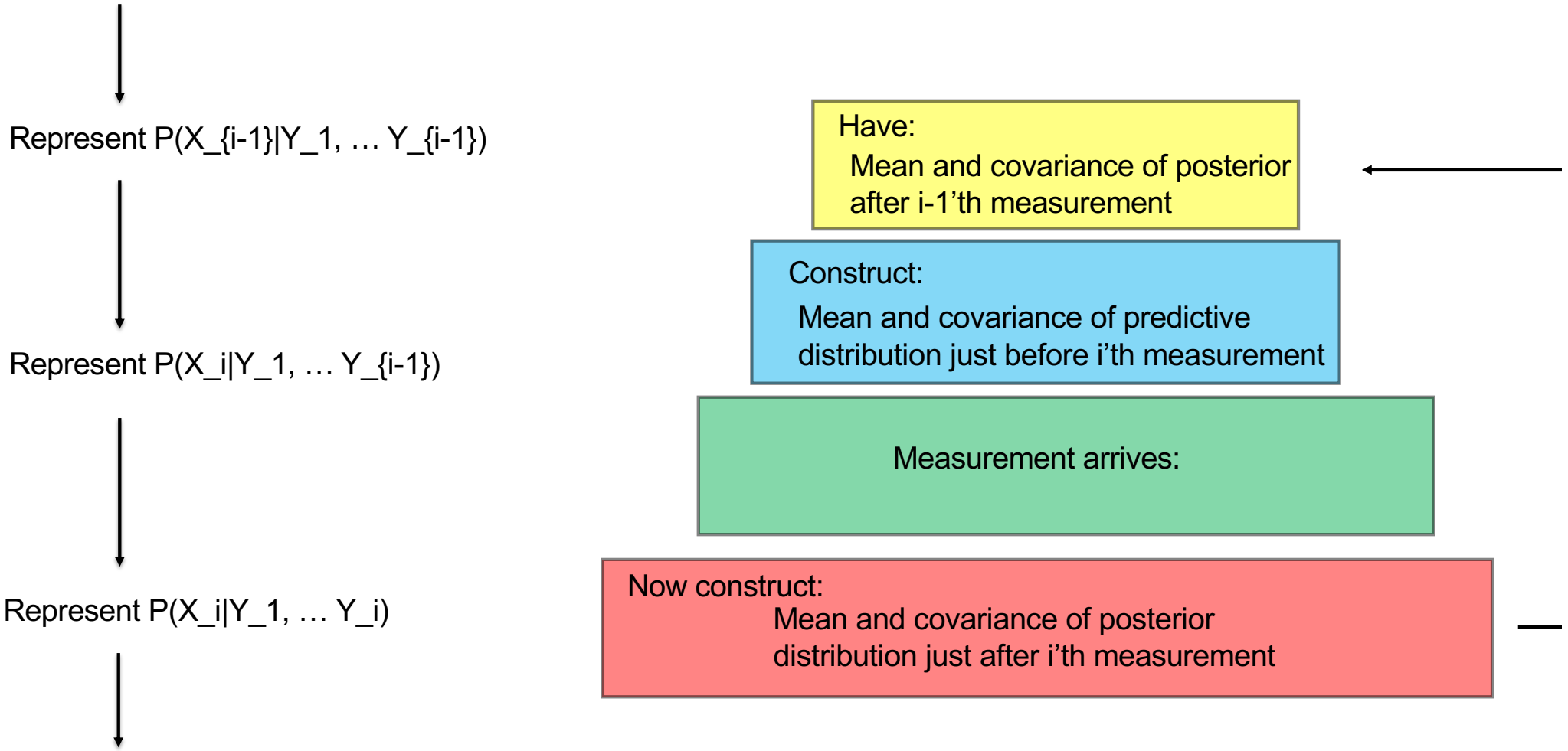
Checking…

- Probability distribution is normal iff it has the form:

$$\log p(\mathbf{x}) = -\frac{1}{2}\left[(\mathbf{x}-\mu)^T \Sigma^{-1}(\mathbf{x}-\mu)\right] + K$$

  - and you can check this for each of the relevant dists.

The steps


Represent P(X_{i-1}|Y_1, … Y_{i-1})
Represent P(X_i|Y_1, … Y_{i-1})
Represent P(X_i|Y_1, … Y_i)
Have:
Mean and covariance of posterior after i-1'th measurement
Construct:
Mean and covariance of predictive distribution just before i'th measurement
Measurement arrives:
Now construct:
Mean and covariance of posterior distribution just after i'th measurement

The Kalman Filter

- Dynamic Model

$$\mathbf{x}_i \sim N(\mathcal{D}_i \mathbf{x}_{i-1}, \Sigma_{d_i})$$
$$\mathbf{y}_i \sim N(\mathcal{M}_i \mathbf{x}_i, \Sigma_{m_i})$$

- Notation

$$\text{mean of } P(X_i|y_0, \ldots, y_{i-1}) \text{ as } \bar{X}_i^-$$
$$\text{mean of } P(X_i|y_0, \ldots, y_i) \text{ as } \bar{X}_i^+$$
$$\text{covar of } P(X_i|y_0, \ldots, y_{i-1}) \text{ as } \Sigma_i^-$$
$$\text{covar of } P(X_i|y_0, \ldots, y_i) \text{ as } \Sigma_i^+$$

Prediction

- We have:

$$\mathbf{x}_{i-1} \sim N(\bar{X}^{+}_{i-1}, \Sigma^{+}_{i-1}) \qquad \mathbf{x}_i \sim N(\mathcal{D}_i \mathbf{x}_{i-1}, \Sigma_{d_i})$$

$$\mathbf{x}_i = \mathcal{D}_i \mathbf{x}_{i-1} + \zeta$$

This is a normal random variable with zero mean and known covariance

$$\text{mean}(\mathbf{x}_i) = \mathcal{D}_i \text{mean}(\mathbf{x}_{i-1})$$

$$\text{cov}(\mathbf{x}_i) = \mathcal{D}_i \text{cov}(\mathbf{x}_{i-1}) \mathcal{D}_i^T + \text{cov}(\zeta)$$

Prediction

- We have:

$$\mathbf{x}_{i-1} \sim N(\bar{X}^{+}_{i-1}, \Sigma^{+}_{i-1}) \qquad \mathbf{x}_i \sim N(\mathcal{D}_i \mathbf{x}_{i-1}, \Sigma_{d_i})$$

$$\mathbf{x}_i = \mathcal{D}_i \mathbf{x}_{i-1} + \zeta$$

This is a normal random variable with zero mean and known covariance

Which yields….

$$\overline{X}^{-}_{i} = \mathcal{D}_i \overline{X}^{+}_{i-1} \qquad \Sigma^{-}_{i} = \Sigma_{d_i} + \mathcal{D}_i \Sigma^{-}_{i-1} \mathcal{D}^{T}_{i}$$

Summary, with change of notation

Useful Fact: 9.2 *The parameters of a normal posterior with a single measurement*

Assume we wish to estimate a parameter θ. The prior distribution for θ is normal, with known mean μ_π and known standard deviation σ_π. We receive a single data item x_1 and a scale c_1. The likelihood of x_1 is normal with mean $c_1\theta$ and standard deviation $\sigma_{m,1}$, where $\sigma_{m,1}$ is known. Then the posterior, $p(\theta|x_1, c_1, \sigma_{m,1}, \mu_\pi, \sigma_\pi)$, is normal, with mean

$$\mu_1 = \frac{c_1 x_1 \sigma_\pi^2 + \mu_\pi \sigma_{m,1}^2}{\sigma_{m,1}^2 + c_1^2 \sigma_\pi^2}$$

posterior mean is weighted combo of prior mean and measurement

and standard deviation

$$\sigma_1 = \sqrt{\frac{\sigma_{m,1}^2 \sigma_\pi^2}{\sigma_{m,1}^2 + c^2 \sigma_\pi^2}}.$$

posterior covar is weighted combo of prior covar, measurement matrix and measurement covar

Useful Fact: 9.2 *The parameters of a normal posterior with a single measurement*

Assume we wish to estimate a parameter θ. The prior distribution for θ is normal, with known mean μ_π and known standard deviation σ_π. We receive a single data item x_1 and a scale c_1. The likelihood of x_1 is normal with mean $c_1\theta$ and standard deviation $\sigma_{m,1}$, where $\sigma_{m,1}$ is known. Then the posterior, $p(\theta|x_1, c_1, \sigma_{m,1}, \mu_\pi, \sigma_\pi)$, is normal, with mean

$$\mu_1 = \frac{c_1 x_1 \sigma_\pi^2 + \mu_\pi \sigma_{m,1}^2}{\sigma_{m,1}^2 + c_1^2 \sigma_\pi^2}$$

and standard deviation

$$\sigma_1 = \sqrt{\frac{\sigma_{m,1}^2 \sigma_\pi^2}{\sigma_{m,1}^2 + c^2 \sigma_\pi^2}}.$$

posterior mean is weighted combo of prior mean and measurement

posterior covar is weighted combo of prior covar, measurement matrix and measurement covar

$$\mathcal{K}_i = \Sigma_i^- \mathcal{M}_i^T \left[\mathcal{M}_i \Sigma_i^- \mathcal{M}_i^T + \Sigma_{m_i} \right]^{-1}$$

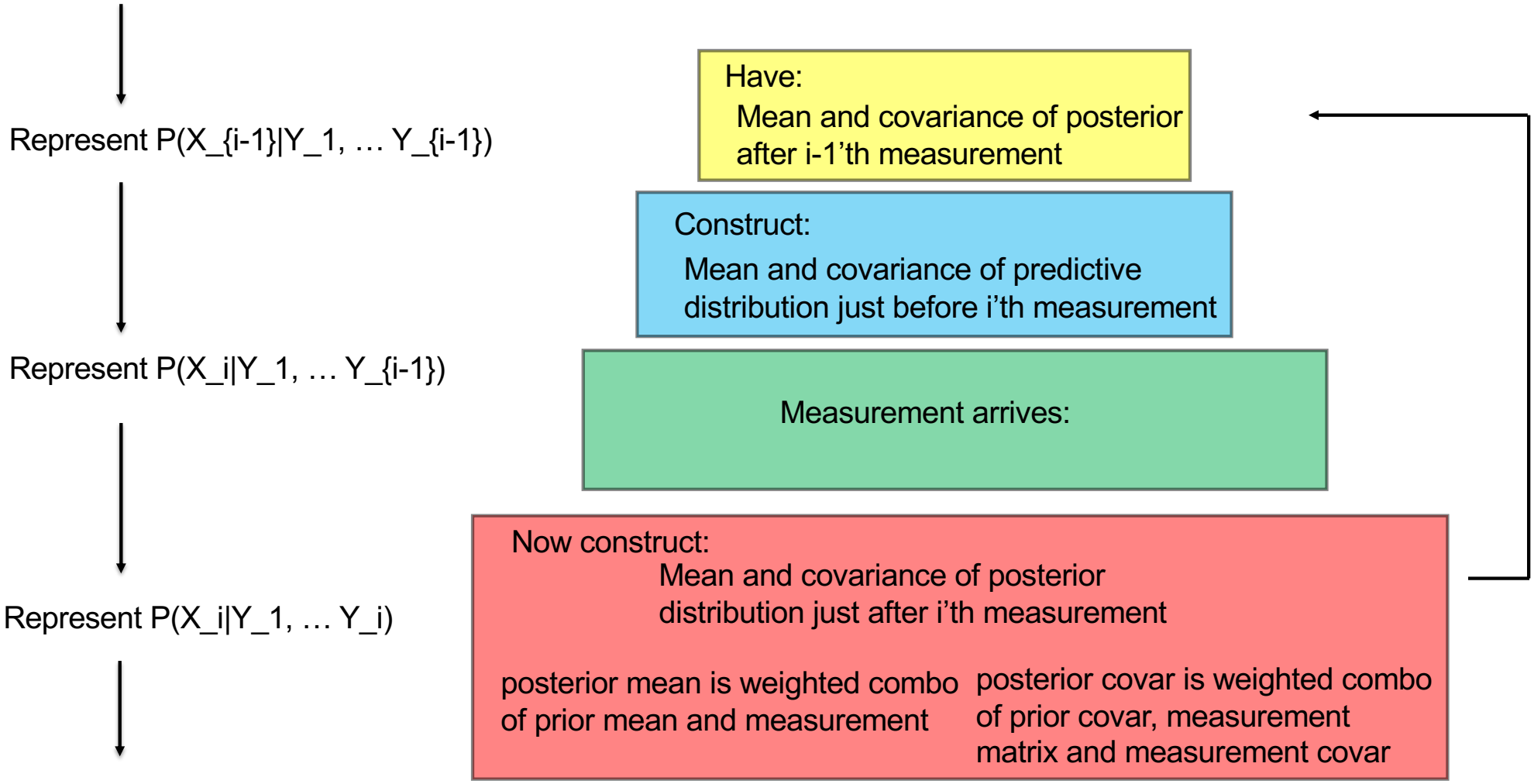
$$\bar{X}_i^+ = \bar{X}_i^- + \mathcal{K}_i \left[\mathbf{y}_i - \mathcal{M}_i \bar{X}_i^- \right]$$

posterior mean is weighted combo of prior mean and measurement

$$\Sigma_i^+ = \left[\mathcal{I} - \mathcal{K}_i \mathcal{M}_i \right] \Sigma_i^-$$

posterior covar is weighted combo of prior covar, measurement matrix and measurement covar

The steps


Represent P(X_{i-1}|Y_1, … Y_{i-1})
Represent P(X_i|Y_1, … Y_{i-1})
Represent P(X_i|Y_1, … Y_i)
Have:
Mean and covariance of posterior after i-1'th measurement
Construct:
Mean and covariance of predictive distribution just before i'th measurement
Measurement arrives:
Now construct:
Mean and covariance of posterior distribution just after i'th measurement
posterior mean is weighted combo of prior mean and measurement
posterior covar is weighted combo of prior covar, measurement matrix and measurement covar

The steps

Represent P(X_{i-1}|Y_1, … Y_{i-1})

Represent P(X_i|Y_1, … Y_{i-1})

Represent P(X_i|Y_1, … Y_i)

Have $\overline{X}^{+}_{i-1}$ Σ^{+}_{i-1}

Construct: $\overline{X}^{-}_{i} = \mathcal{D}_i \overline{X}^{+}_{i-1}$

$$\Sigma^{-}_{i} = \Sigma_{d_i} + \mathcal{D}_i \Sigma^{-}_{i-1} \mathcal{D}^{T}_{i}$$

Measurement arrives:

$$\mathbf{y}_i \sim N(\mathcal{M}_i \mathbf{x}_i; \Sigma_{mi})$$

Now construct: $\mathcal{K}_i = \Sigma^{-}_{i} \mathcal{M}^{T}_{i} \left[\mathcal{M}_i \Sigma^{-}_{i} \mathcal{M}^{T}_{i} + \Sigma_{m_i} \right]^{-1}$

$$\bar{X}^{+}_{i} = \bar{X}^{-}_{i} + \mathcal{K}_i \left[\mathbf{y}_i - \mathcal{M}_i \bar{X}^{-}_{i} \right]$$

$$\Sigma^{+}_{i} = \left[\mathcal{I} - \mathcal{K}_i \mathcal{M}_i \right] \Sigma^{-}_{i}$$

Simple example: tracking a ballistic object

Assumptions:

- Orthographic image, camera oriented so vertical (y) is gravity
- Object is falling freely under gravity
- Detector measures object position + noise

The object

State:

position, velocity, acceleration

$$\mathbf{x} = \begin{pmatrix} \mathbf{p} \\ \mathbf{v} \\ \mathbf{a} \end{pmatrix}$$

$$\begin{aligned} \mathbf{p}_i &= \mathbf{p}_{i-1} + \mathbf{v}_{i-1}\Delta t + \eta_{i-1} \\ \mathbf{v}_i &= \mathbf{v}_{i-1} + \mathbf{a}_{i-1}\Delta t + \zeta_{i-1} \\ \mathbf{a}_i &= \mathbf{a}_{i-1} + \xi_{i-1} \end{aligned}$$

$$\begin{pmatrix} \mathbf{p}_i \\ \mathbf{v}_i \\ \mathbf{a}_i \end{pmatrix} = \begin{pmatrix} \mathcal{I} & \Delta t\mathcal{I} & 0 \\ 0 & \mathcal{I} & \Delta t\mathcal{I} \\ 0 & 0 & \mathcal{I} \end{pmatrix} \begin{pmatrix} \mathbf{p}_{i-1} \\ \mathbf{v}_{i-1} \\ \mathbf{a}_{i-1} \end{pmatrix} + \epsilon_i$$

The object

Prior:

position and velocity could be pretty much anything

acceleration should be very close to gravity vector

Mean: $\begin{pmatrix} 0 \\ 0 \\ \mathbf{g} \end{pmatrix}$

Covariance $\begin{pmatrix} w\mathcal{I} & 0 & 0 \\ 0 & w\mathcal{I} & 0 \\ 0 & 0 & s\mathcal{I} \end{pmatrix}$

Where w is big and s is small

Measurements

At i'th frame, we see:

x and z components of position + noise

So

$$\mathbf{y}_i = \begin{pmatrix} p_1 \\ p_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \mathbf{x}_i$$

Choose noise model, and we are done!

The steps

Represent P(X_{i-1}|Y_1, … Y_{i-1})

Represent P(X_i|Y_1, … Y_{i-1})

Represent P(X_i|Y_1, … Y_i)

Have $\overline{X}^{+}_{i-1}$ Σ^{+}_{i-1}

Construct: $\overline{X}^{-}_{i} = \mathcal{D}_i \overline{X}^{+}_{i-1}$

$$\Sigma^{-}_{i} = \Sigma_{d_i} + \mathcal{D}_i \Sigma^{-}_{i-1} \mathcal{D}^{T}_{i}$$

Measurement arrives:

$$\mathbf{y}_i \sim N(\mathcal{M}_i \mathbf{x}_i; \Sigma_{mi})$$

Now construct: $\mathcal{K}_i = \Sigma^{-}_{i} \mathcal{M}^{T}_{i} \left[\mathcal{M}_i \Sigma^{-}_{i} \mathcal{M}^{T}_{i} + \Sigma_{m_i} \right]^{-1}$

$$\bar{X}^{+}_{i} = \bar{X}^{-}_{i} + \mathcal{K}_i \left[\mathbf{y}_i - \mathcal{M}_i \bar{X}^{-}_{i} \right]$$

$$\Sigma^{+}_{i} = \left[\mathcal{I} - \mathcal{K}_i \mathcal{M}_i \right] \Sigma^{-}_{i}$$

Questions:

Can we estimate p_2 like this?

Why can we estimate velocity and acceleration?

Why go to all this trouble?

The steps

Represent P(X_{i-1}|Y_1, … Y_{i-1})

Represent P(X_i|Y_1, … Y_{i-1})

Represent P(X_i|Y_1, … Y_i)

Have $\overline{X}^{+}_{i-1}$ Σ^{+}_{i-1}

Construct: $\overline{X}^{-}_{i} = \mathcal{D}_i \overline{X}^{+}_{i-1}$

$$\Sigma^{-}_{i} = \Sigma_{d_i} + \mathcal{D}_i \Sigma^{-}_{i-1} \mathcal{D}^{T}_{i}$$

Measurement arrives:

$$\mathbf{y}_i \sim N(\mathcal{M}_i \mathbf{x}_i; \Sigma_{mi})$$

Now construct: $\mathcal{K}_i = \Sigma^{-}_{i} \mathcal{M}^{T}_{i} \left[\mathcal{M}_i \Sigma^{-}_{i} \mathcal{M}^{T}_{i} + \Sigma_{m_i} \right]^{-1}$

$$\bar{X}^{+}_{i} = \bar{X}^{-}_{i} + \mathcal{K}_i \left[\mathbf{y}_i - \mathcal{M}_i \bar{X}^{-}_{i} \right]$$

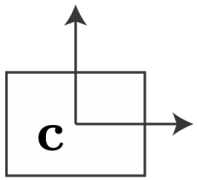
$$\Sigma^{+}_{i} = \left[\mathcal{I} - \mathcal{K}_i \mathcal{M}_i \right] \Sigma^{-}_{i}$$

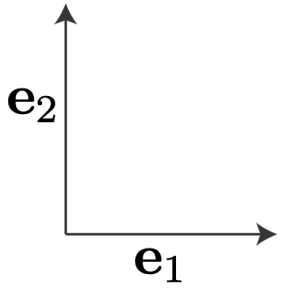
Example II:

- A simple translating car
  - we supply a known demand to the accelerator,
    - changing at each time step
  - it sees 2 beacons (which are in its coordinate system)
    - beacon 1 measured in car x but not y
    - beacon 2 measured in car y but not x
- Q:
  - recover filtered estimates of:
    - position, velocity and acceleration in world coords

$\mathbf{b}_2$ ○

$\mathbf{b}_1$ ○





In world coordinates, car is at: $\mathbf{c}$

In car coordinates, beacon 1 measurement is: $\mathbf{e}_1^T(\mathbf{b}_1 - \mathbf{c})$

In car coordinates, beacon 2 measurement is: $\mathbf{e}_2^T(\mathbf{b}_2 - \mathbf{c})$

Dynamical model

- We supply a demand to the accelerator
  - acceleration updates as noise (measured to be about the same as demand!)

$$\mathbf{a}_{i+1} = \mathbf{a}_i + \text{noise}$$

  - velocity by integrating acceleration

$$\mathbf{v}_{i+1} = \mathbf{v}_i + \delta t \mathbf{a}_i + \text{noise}$$

  - position by integrating velocity

$$\mathbf{c}_{i+1} = \mathbf{c}_i + \delta t \mathbf{v}_i + \text{noise}$$

Stack the vectors to get:

$$\mathbf{x}_i = \begin{bmatrix} \mathbf{c}_i \\ \mathbf{v}_i \\ \mathbf{a}_i \end{bmatrix}$$

Which gives:

$$\mathbf{x}_{i+1} = \begin{bmatrix} \mathbf{c}_{i+1} \\ \mathbf{v}_{i+1} \\ \mathbf{a}_{i+1} \end{bmatrix} = \begin{bmatrix} \mathcal{I} & \delta t\mathcal{I} & 0 \\ 0 & \mathcal{I} & \delta t\mathcal{I} \\ 0 & 0 & \mathcal{I} \end{bmatrix} \begin{bmatrix} \mathbf{c}_i \\ \mathbf{v}_i \\ \mathbf{a}_i \end{bmatrix} + \xi_i$$

Where:

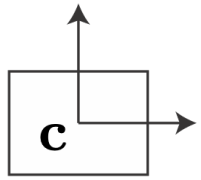
$$\xi_i \sim N(\mathbf{0}; \Sigma_{d,i})$$

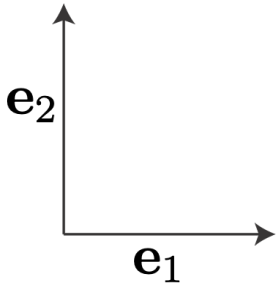
Measurement model

- The acceleration at i should be demand
  - +noise
- Beacons are in car coordinate system)
  - beacon 1 measured in car x but not y
  - beacon 2 measured in car y but not x

$\mathbf{b}_2$

$\mathbf{b}_1$





In world coordinates, car is at: $\mathbf{c}$

In car coordinates, beacon 1 measurement is: $\mathbf{e}_1^T(\mathbf{b}_1 - \mathbf{c})$

In car coordinates, beacon 2 measurement is: $\mathbf{e}_2^T(\mathbf{b}_2 - \mathbf{c})$

The acceleration demand

$$\mathbf{y}_i = \begin{bmatrix} \mathbf{d}_i \\ \mathbf{e}_1^T \mathbf{b}_1 - b_1 \\ \mathbf{e}_2^T \mathbf{b}_2 - b_2 \end{bmatrix} + \text{noise}$$

These are known constants Measurements from the beacons

$$\mathbf{y}_i = \begin{bmatrix} 0 & 0 & \mathcal{I} \\ \mathbf{e}_1^T & 0 & 0 \\ \mathbf{e}_2^T & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{c}_i \\ \mathbf{v}_i \\ \mathbf{a}_i \end{bmatrix} + \text{noise} = \begin{bmatrix} 0 & 0 & \mathcal{I} \\ \mathbf{e}_1^T & 0 & 0 \\ \mathbf{e}_2^T & 0 & 0 \end{bmatrix} \mathbf{x}_i + \zeta_i$$

$$\zeta_i \sim N(0; \Sigma_{m,i})$$

The steps

Represent P(X_{i-1}|Y_1, … Y_{i-1})

Represent P(X_i|Y_1, … Y_{i-1})

Represent P(X_i|Y_1, … Y_i)

Have $\overline{X}^{+}_{i-1}$ Σ^{+}_{i-1}

Construct: $\overline{X}^{-}_{i} = \mathcal{D}_i \overline{X}^{+}_{i-1}$

$$\Sigma^{-}_{i} = \Sigma_{d_i} + \mathcal{D}_i \Sigma^{-}_{i-1} \mathcal{D}^{T}_{i}$$

Measurement arrives:

$$\mathbf{y}_i \sim N(\mathcal{M}_i \mathbf{x}_i; \Sigma_{mi})$$

Now construct: $\mathcal{K}_i = \Sigma^{-}_{i} \mathcal{M}^{T}_{i} \left[\mathcal{M}_i \Sigma^{-}_{i} \mathcal{M}^{T}_{i} + \Sigma_{m_i} \right]^{-1}$

$$\bar{X}^{+}_{i} = \bar{X}^{-}_{i} + \mathcal{K}_i \left[\mathbf{y}_i - \mathcal{M}_i \bar{X}^{-}_{i} \right]$$

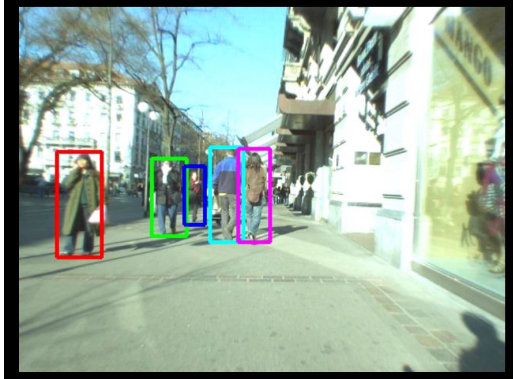
$$\Sigma^{+}_{i} = \left[\mathcal{I} - \mathcal{K}_i \mathcal{M}_i \right] \Sigma^{-}_{i}$$

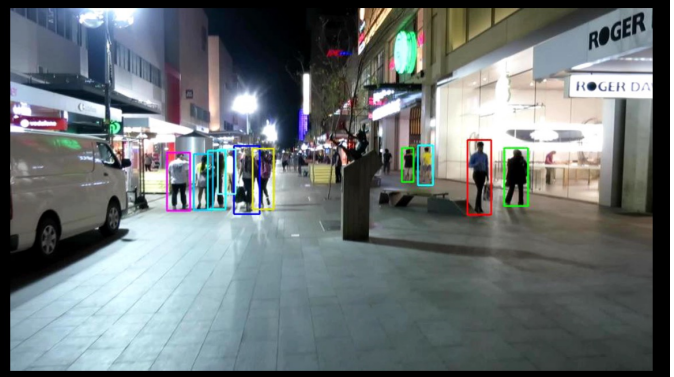
Questions:

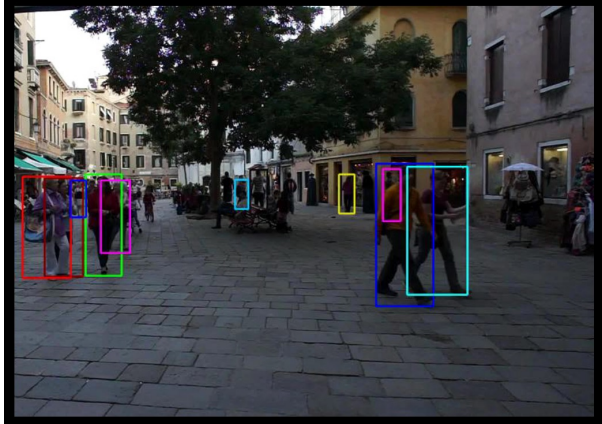
Why can we estimate position and velocity?

Why go to all this trouble?

Multi-Object Tracking





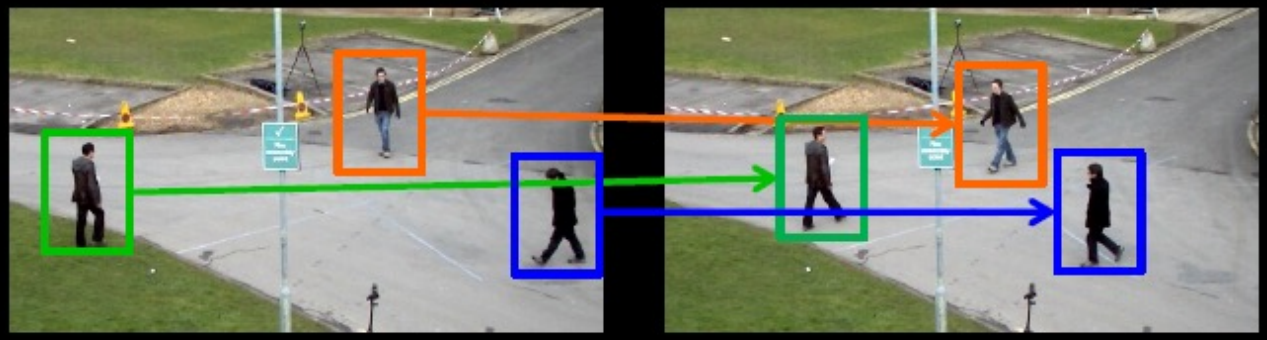


Goal: detect and track all objects in a scene

Multiple slides lifted from L. Leal-Taixe's slides at ICVSS 2022

Problem Statement

- Given a video, find out which parts of the image depict the same object in different frames
- Often we use detectors as starting points



TRACKING IS ALSO…

- Learning to model our target:

- **Appearance**: we need to know how the target looks like
  - Single object tracking
  - Re-identification

- **Motion**: to make predictions of where the targets goes
  - Trajectory prediction

CHALLENGES

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

OFFLINE VS. ONLINE

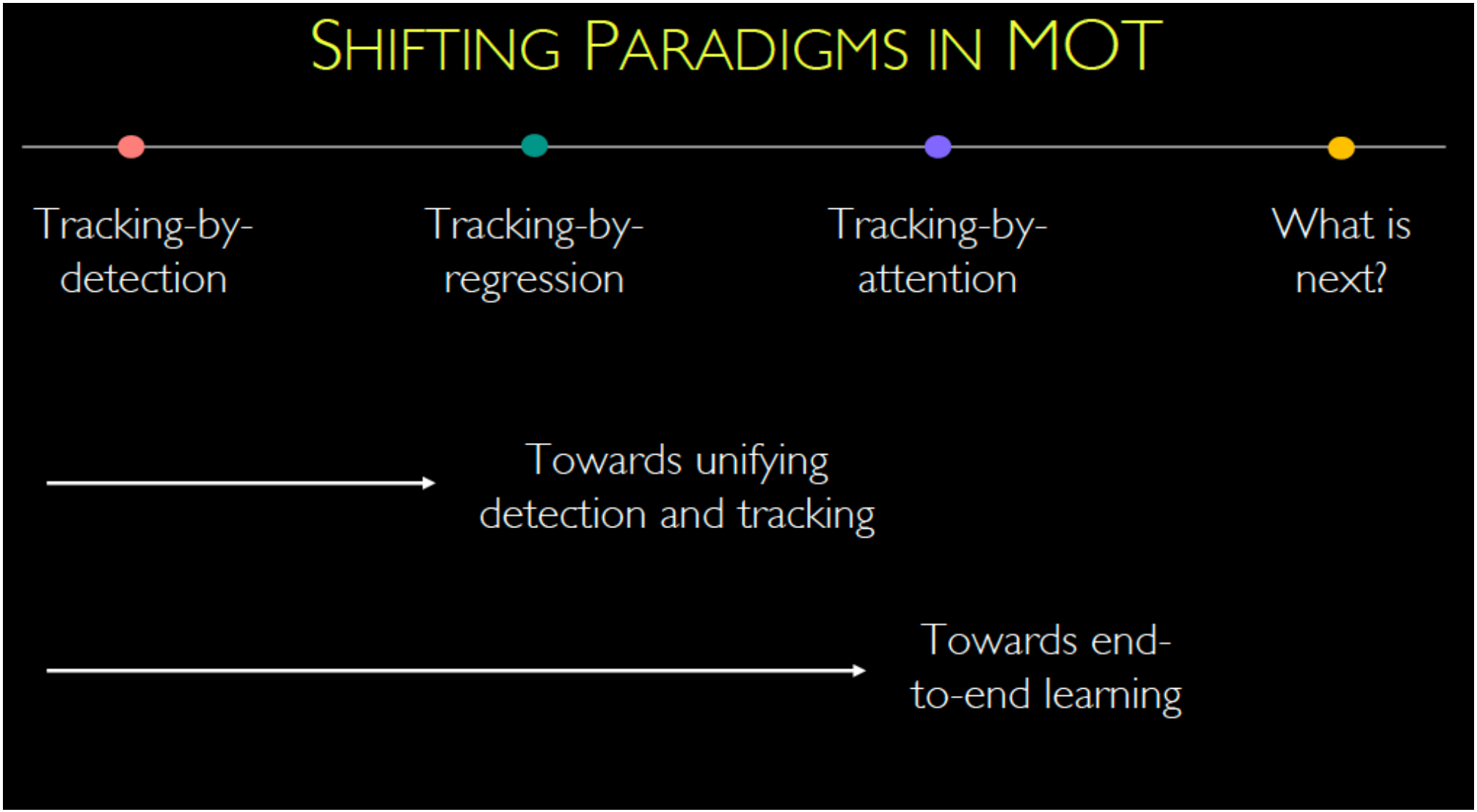
- Online tracking
  - Processes frames as they become available
  - For real-time applications, e.g., autonomous driving, AR/VR
  - Prone to drifting → hard to recover from errors or occlusion

Sometimes “causal”

- 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, automatic labeling, video editing.

Offline vs. Online

- Online tracking
  - Tracking-by-regression, e.g., Tracktor, Centertrack.
  - Transformer-based trackers, e.g., Trackformer
- Offline tracking
  - Tracking with graphical models
  - Learning to track with graph neural networks, e.g., MPNTrack


SHIFTING PARADIGMS IN MOT
Tracking-by-detection
Tracking-by-regression
Tracking-by-attention
What is next?
Towards unifying detection and tracking
Towards end-to-end learning

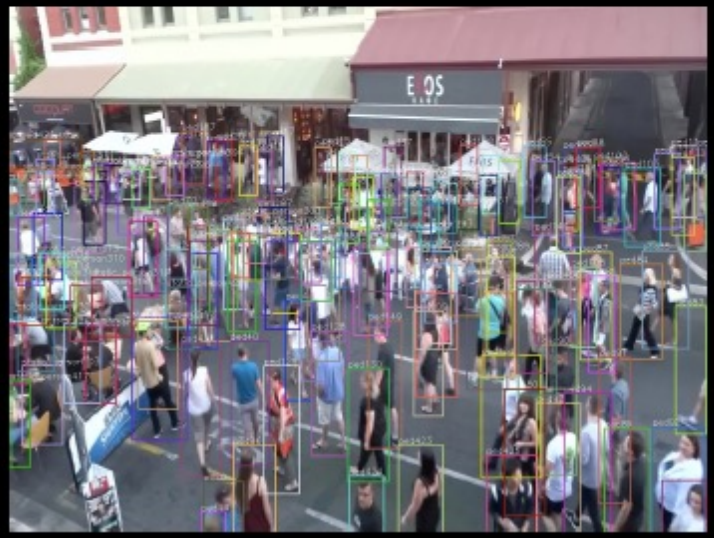
MOTCHALLENGE

- MOTChallenge: www.motchallenge.net
  - Multiple object tracking (from sparse to extremely crowded)

MOT15

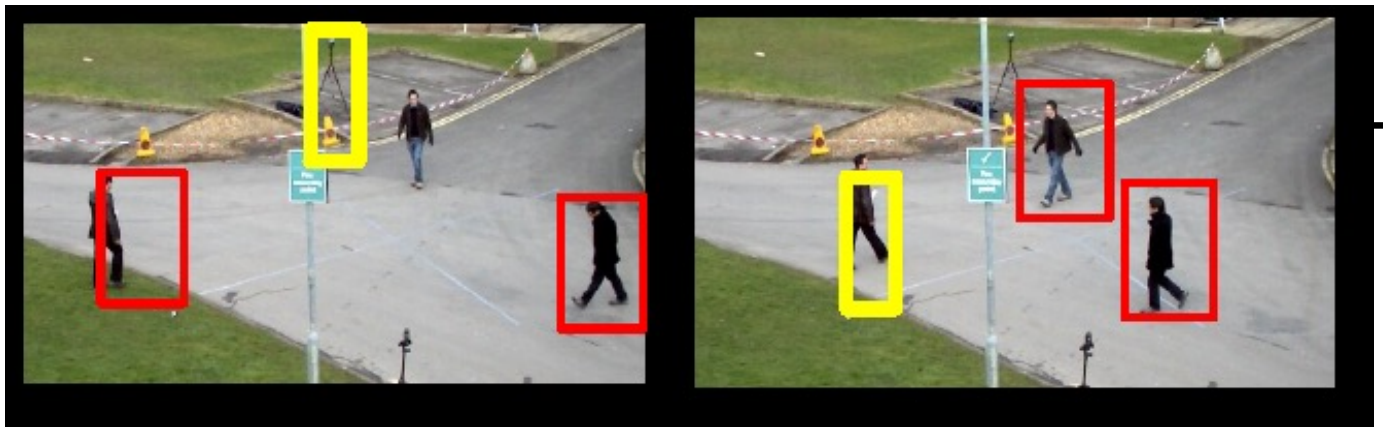
MOT16/17

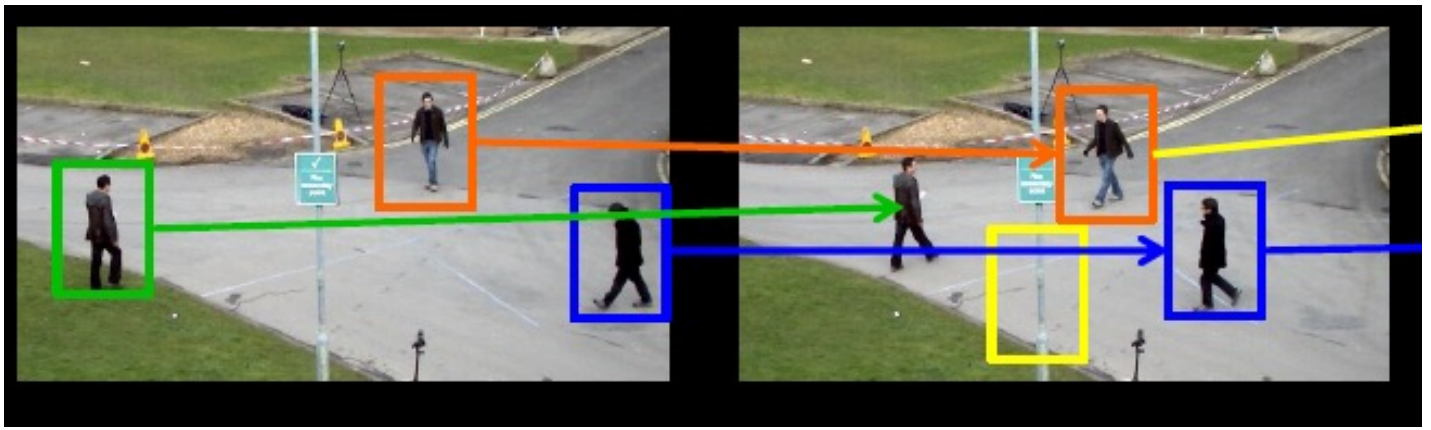
MOT20



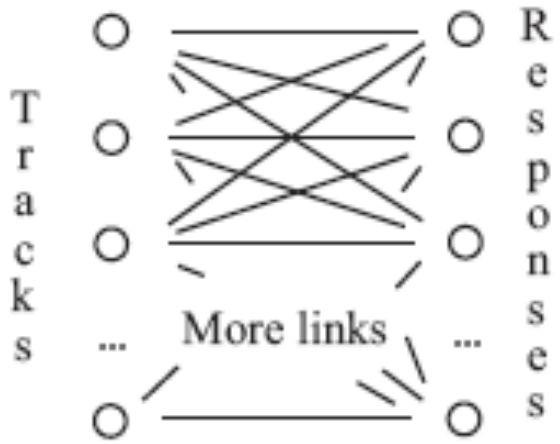
Dendorfer et al. *MOTChallenge: A benchmark for single-camera multi-target tracking.* IJCV, 2021.

Tracking-by-Detection

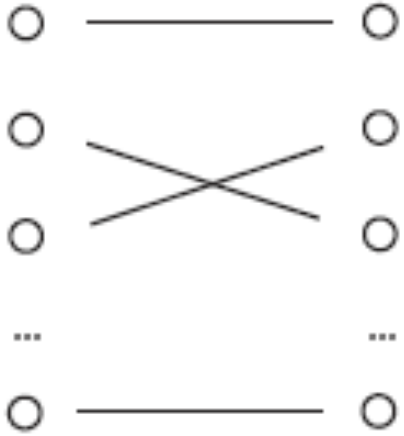




TBD (to date)



Weighted bipartite matching

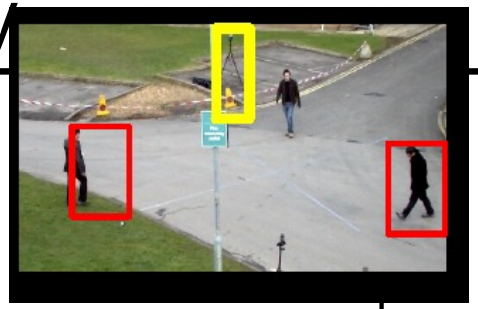


Questions:

can we use dynamics to simplify matching?

how do we represent similarity?

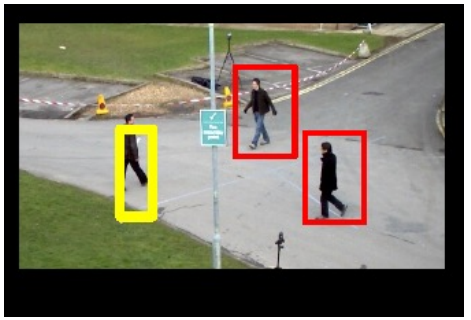
Recall Kalman Filter story



Have:
Mean and covariance of posterior after i-1'th measurement

Construct:
Mean and covariance of predictive distribution just before i'th measurement

This predicts where Boxes might be in Next frame



Measurement arrives:

Now construct:
Mean and covariance of posterior distribution just after i'th measurement

posterior mean is weighted combo of prior mean and measurement

A Simple Online Tracker

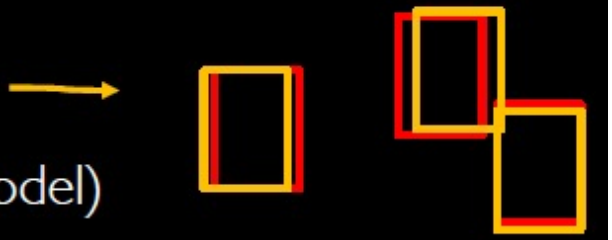
t

t+1

t+2

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

- 2. Prediction of the next position (motion model)



- 3. **Matching** predictions with detections (appearance model)

A Simple Online Tracker

- 2. Prediction of the next position (motion model)
  - Classic: Kalman filter
  - Nowadays: Recurrent architecture Builld this into detector
  - For now: we will assume a constant velocity model (spoiler alter: it works really well at high framerates and without occlusions!)

Improvements

Adjust detector to predict next object location

Learn feature representations of boxes that get the tracks right

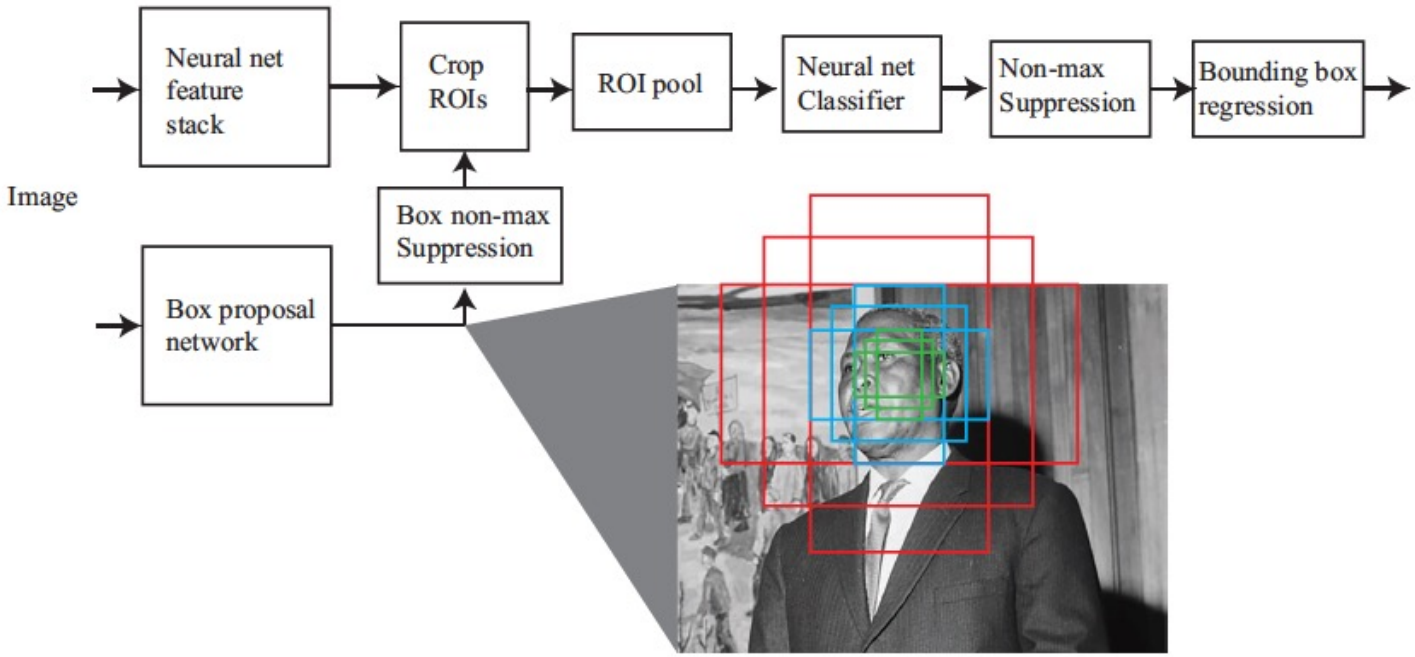
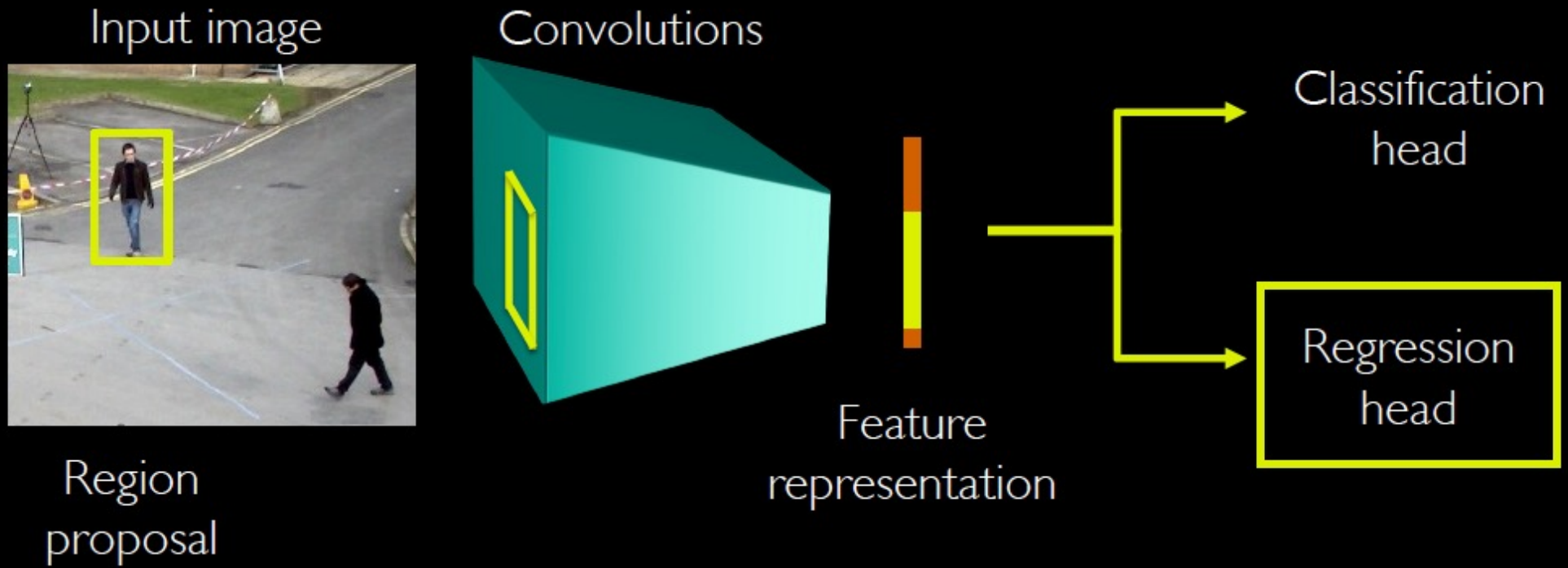


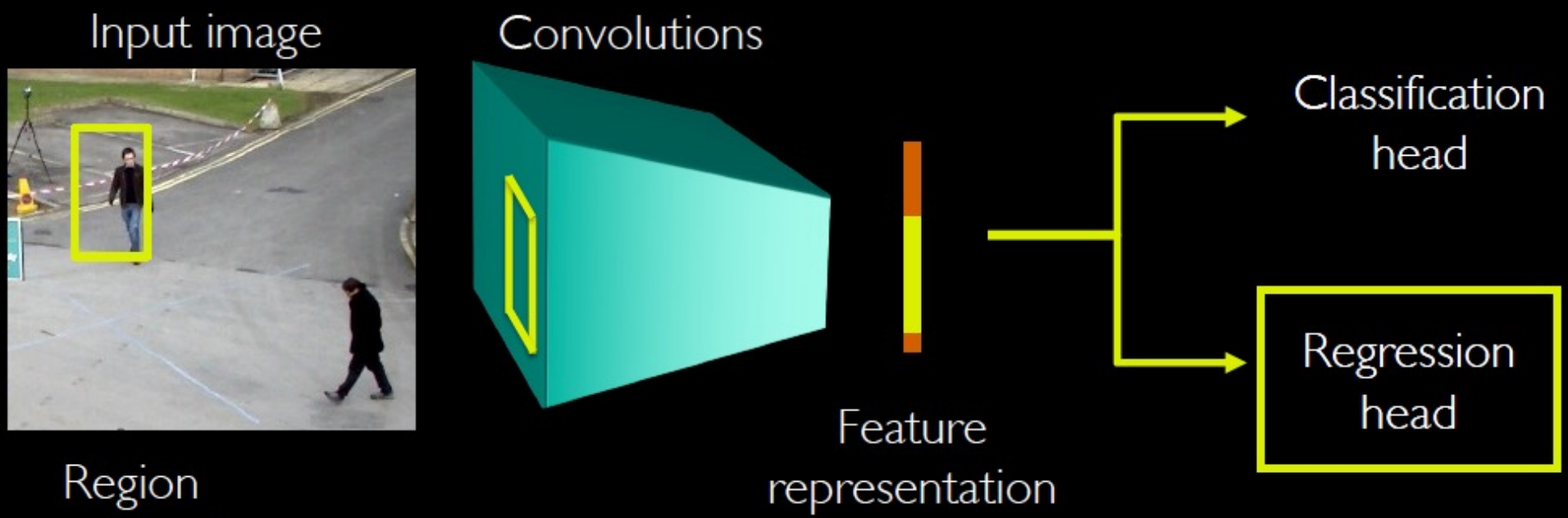
FIGURE 18.8: *Faster RCNN uses two networks. One uses the image to compute "objectness" scores for a sampling of possible image boxes. The samples (called "anchor boxes") are each centered at a grid point. At each grid point, there are nine boxes (three scales, three aspect ratios). The second is a feature stack that computes a representation of the image suitable for classification. The boxes with highest objectness score are then cut from the feature map, standardized with ROI pooling, then passed to a classifier. Bounding box regression means that the relatively coarse sampling of locations, scales and aspect ratios does not weaken accuracy.*

Regression-Based Detectors



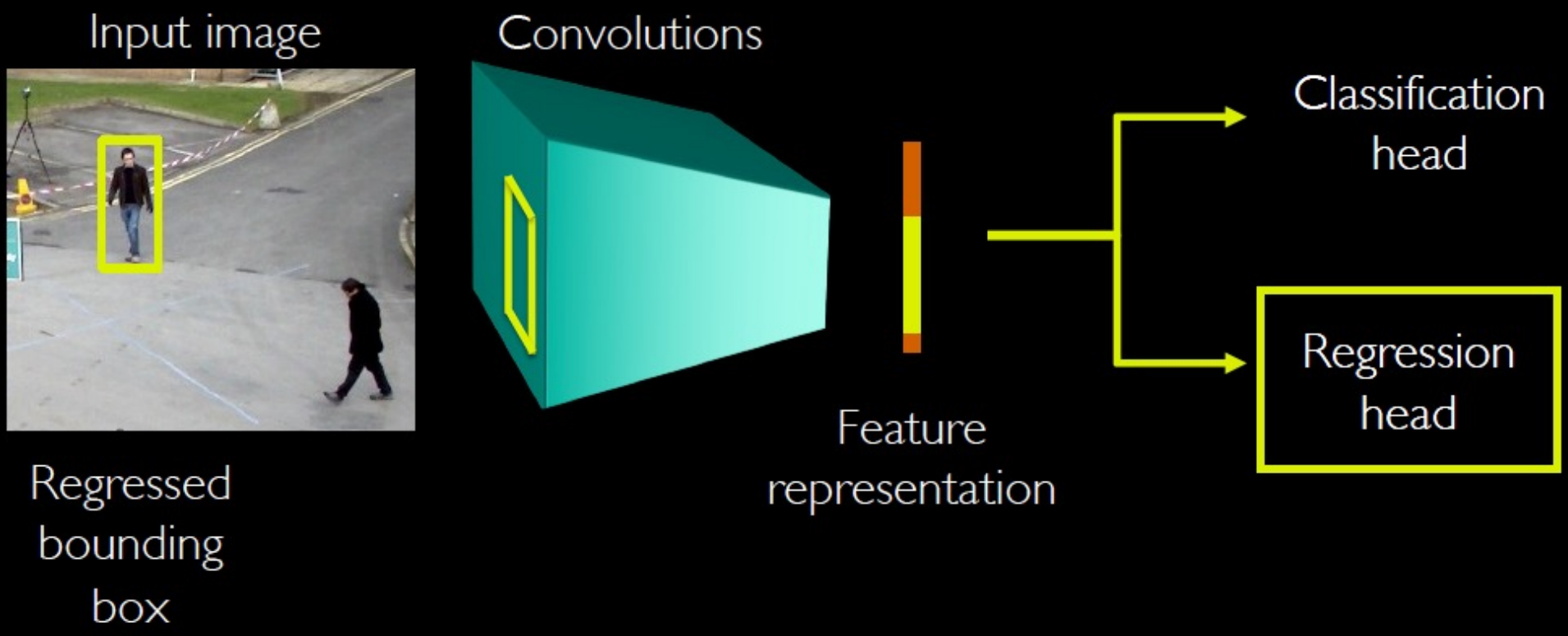
R. Girshick. *Faster RCNN*. ICCV 2015

REGRESSION-BASED DETECTORS



R. Girshick. *Faster RCNN*. ICCV 2015

Regression-Based Detectors



R. Girshick. *Faster RCNN*. ICCV 2015

From Detector to Tracktor

- This is very similar to what we want to do in online tracking

- Tracktor: a method trained as a detector but with tracking capabilities

Bergmann et al. *Tracking without bells and whistles*. ICCV 2019.

From Detector to Tracktor

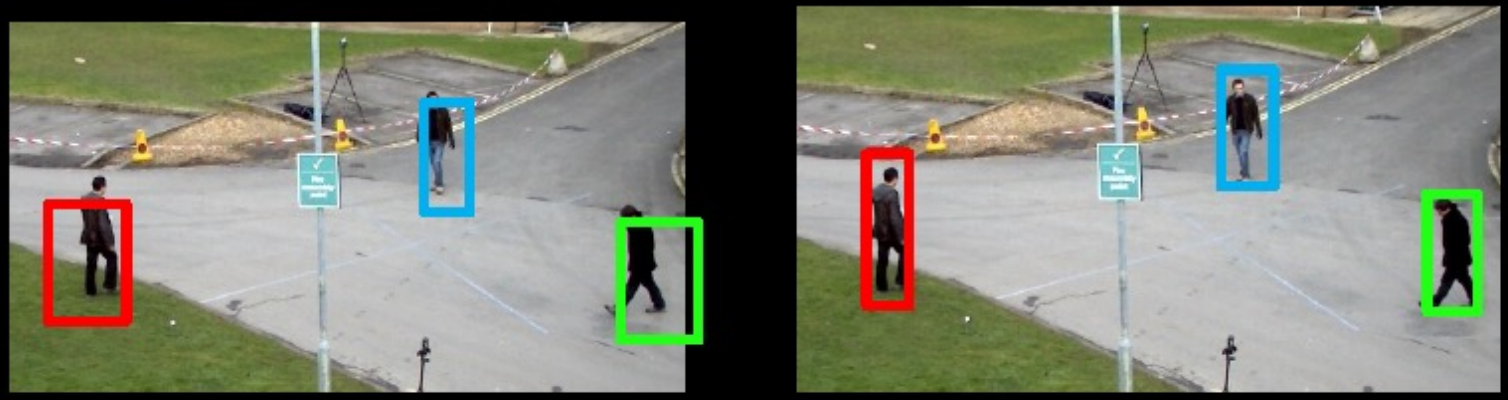
Frame t+1

Use detections of frame t as proposals

Bergmann et al. *Tracking without bells and whistles*. ICCV 2019.

From Detector to Tracktor

Bounding box regression



Where did the detection with ID 1 go in the next frame? ✓ Tracking!

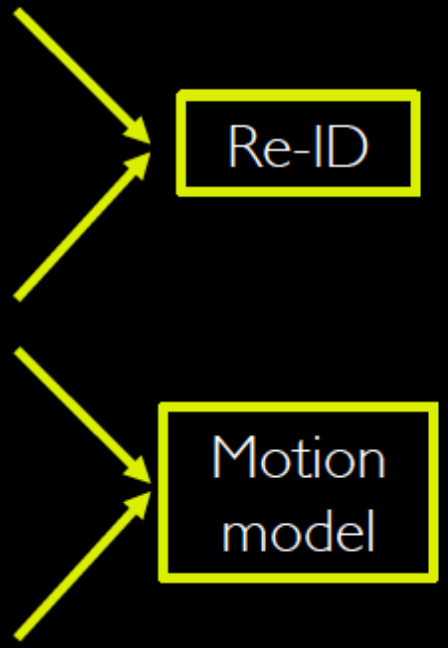
Bergmann et al. *Tracking without bells and whistles.* ICCV 2019.

Pros and Cons

- PRO We can reuse an extremely well-trained regressor
  - We get well-positioned bounding boxes
- PRO We can train our model on still images → easier annotation!
- PRO Tracktor is online

Pros and Cons

- CON There is no notion of "identity" in the model
  - Confusion in crowded spaces
- CON As any online tracker, the track is killed if the target becomes occluded
  - Need to close small gaps and occlusions
- CON The regressor only shifts the box by a small quantity
  - Large camera motions
  - Large displacements due to low framerate



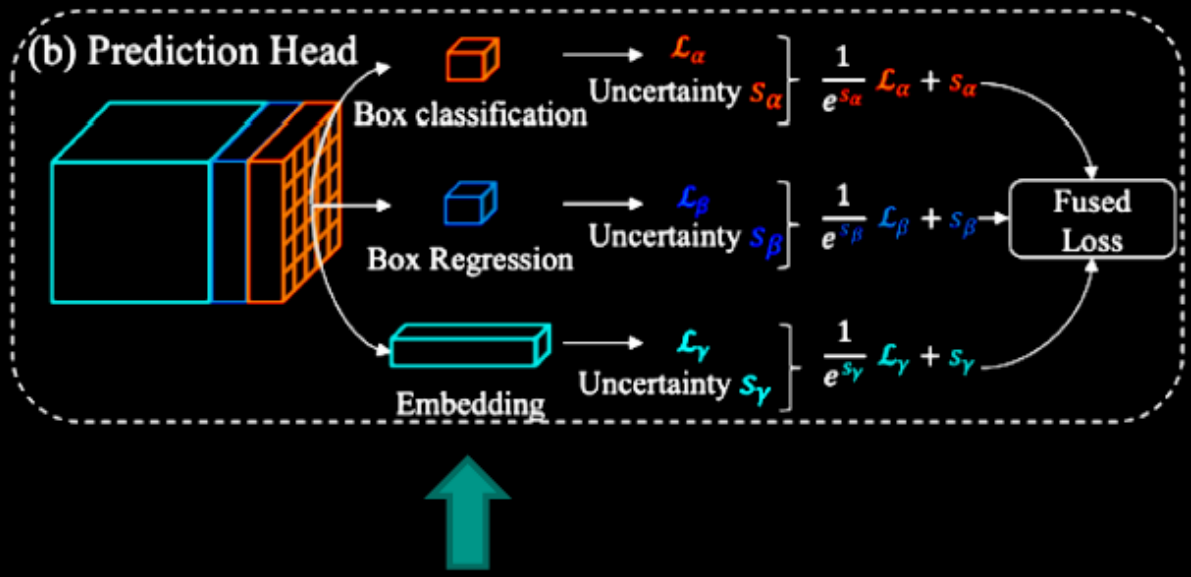
Improvements

Adjust detector to predict next object location

Learn feature representations of boxes that get the tracks right

TOWARDS UNIFYING DETECTION AND TRACKING

- Option 1: Joint detection and association embedding prediction (JDE)



Wang et al. *Towards real-time multi-object tracking.* ECCV 2020.
Xu et al. *How to train your deep multi-object tracker.* CVPR 2020

Simplest version

Train so that

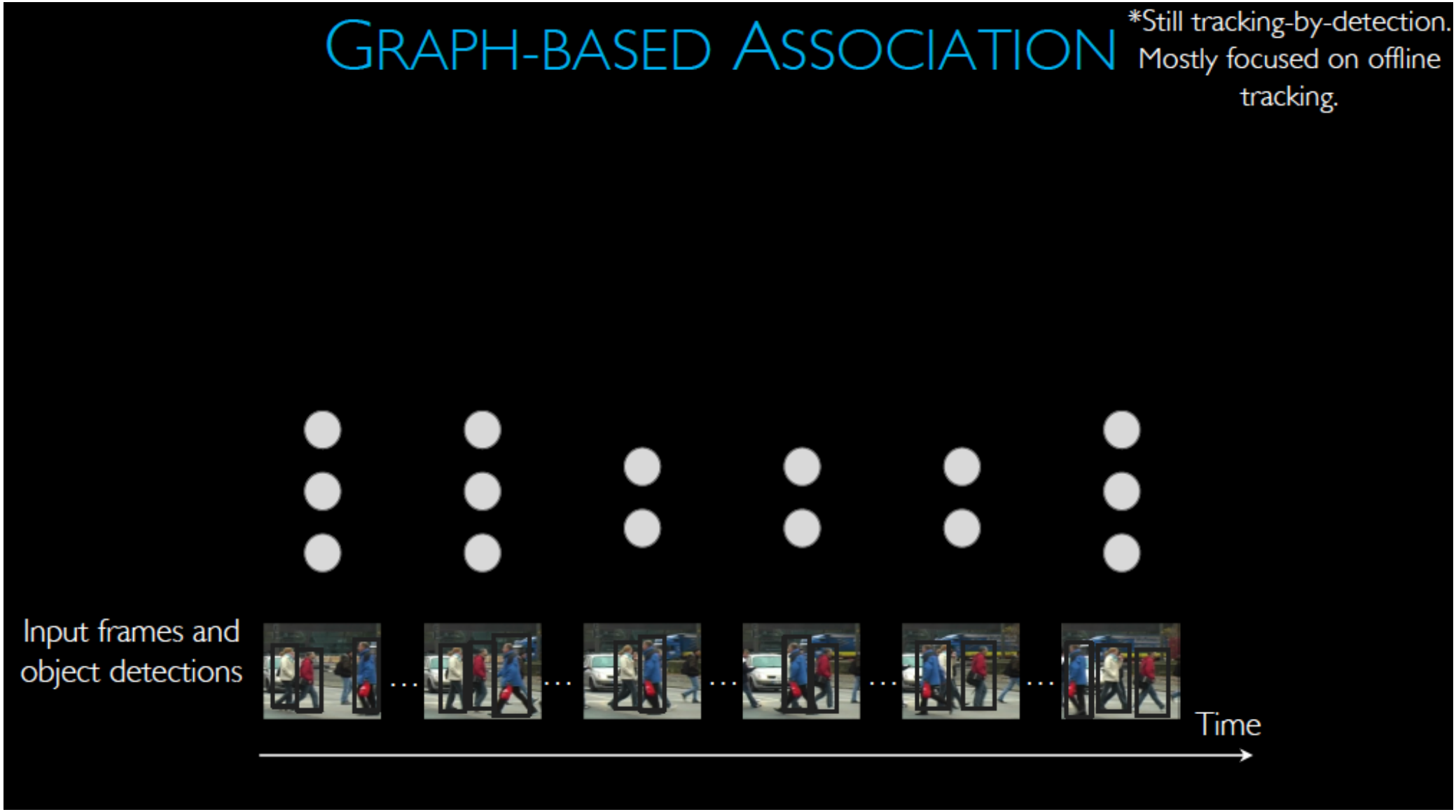
- embeddings for correct matches are similar
- embeddings for wrong matches are different

More complicated

- set up long term matching process (Graph neural network)

GRAPH-BASED ASSOCIATION

*Still tracking-by-detection. Mostly focused on offline tracking.



Graph-based Association

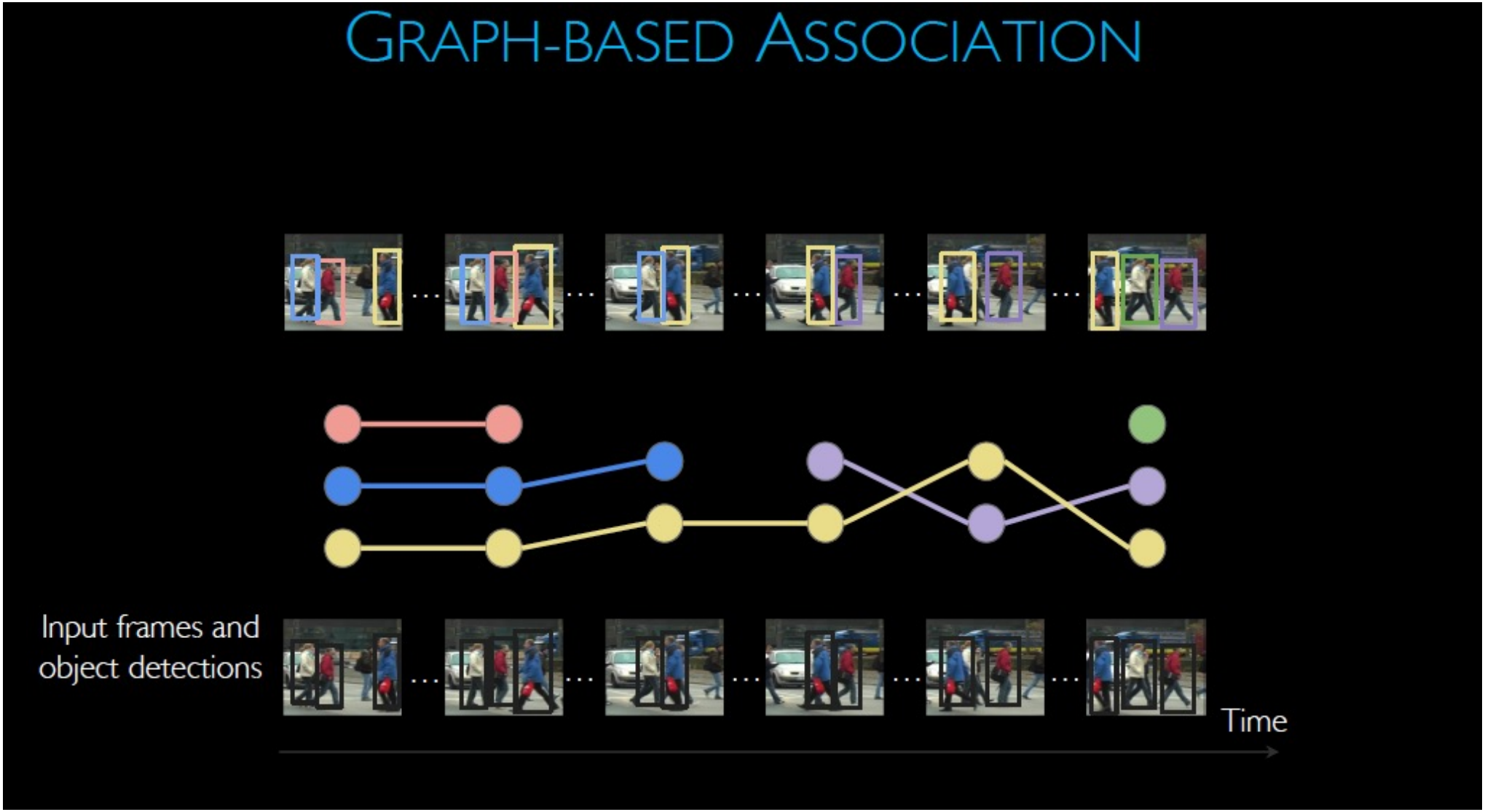
- Pairwise edge costs can either be learned or handcrafted (same as for the Hungarian)
- Find trajectories with a solver, e.g., Simplex

Input frames and object detections

Time

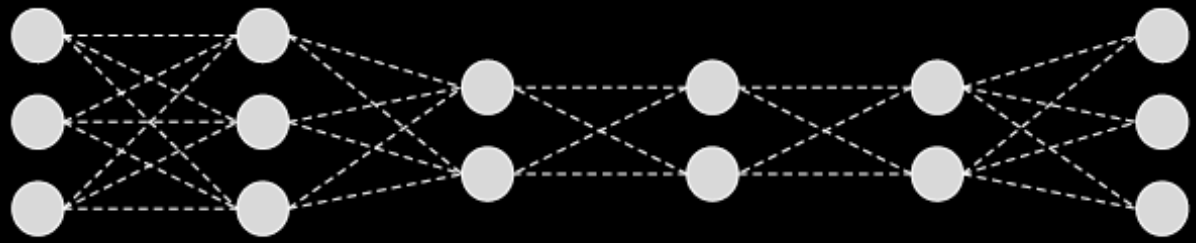
Zhang et al. *Global Data Association for Multi-Object Tracking Using Network Flows*. CVPR 2008

GRAPH-BASED ASSOCIATION



Graph-based Association

- ✗ Feature extraction is done independently from the optimization problem
- ✗ Optimization can be expensive (depends on the graph connectivity)

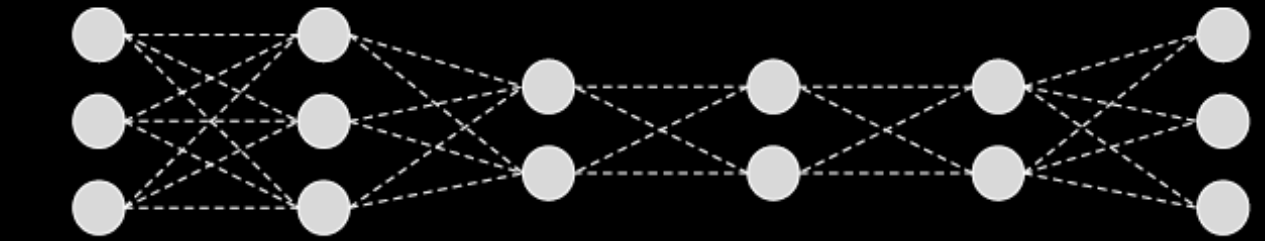


Input frames and object detections

Time

GNN-BASED ASSOCIATION

- Solution: more machine learning!

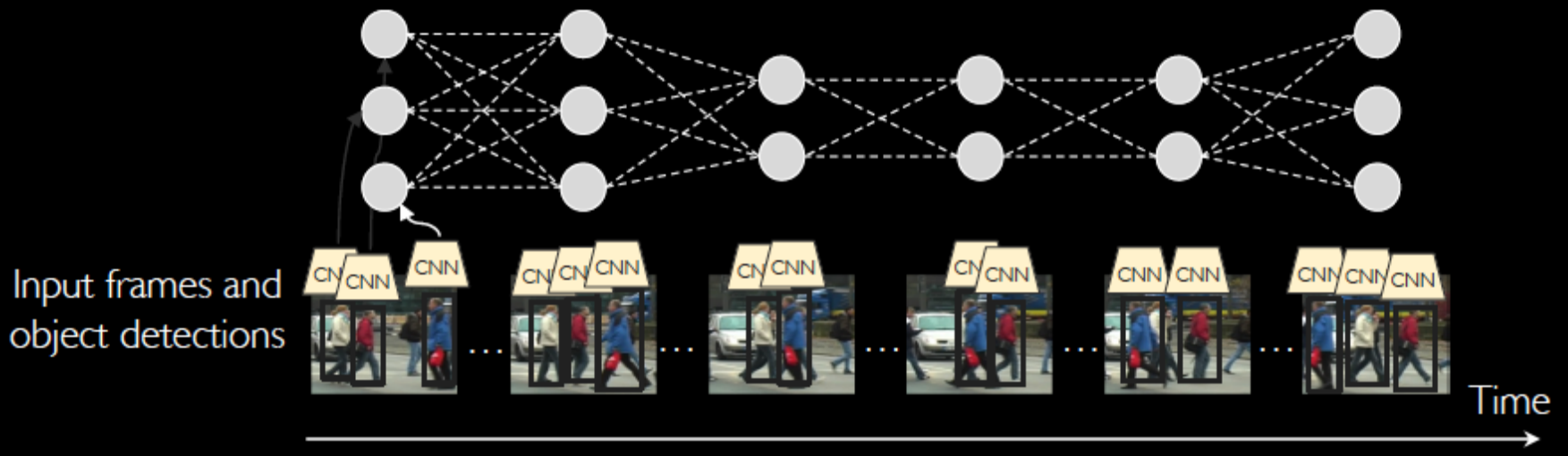


Input frames and object detections

Time

GNN-BASED ASSOCIATION

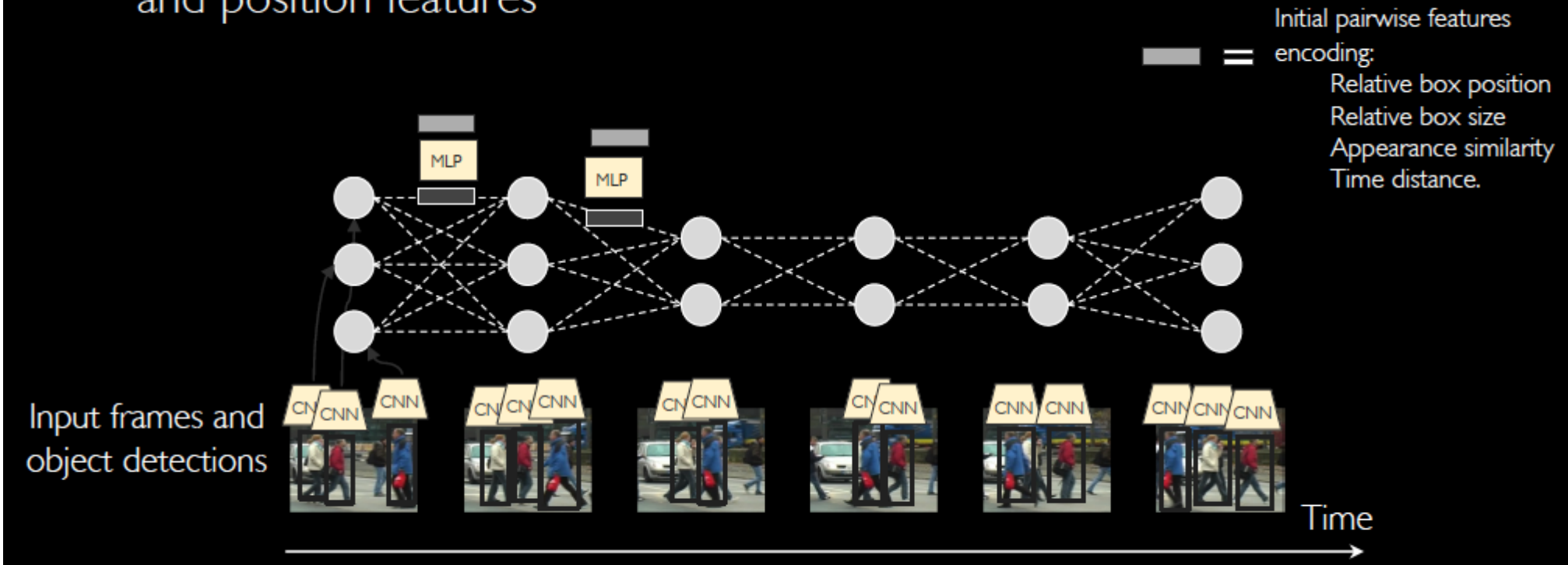
- Node embeddings are obtained from a CNN



Leal-Taixé et al. Learning by tracking: Siamese CNN for robust target association. CVPRW 2016

GNN-BASED ASSOCIATION

- Node embeddings are obtained from a CNN
- Edge embeddings are obtained from an MLP operating on appearance and position features



G. Brasó and L. Leal-Taixé. *Learning a neural solver for multiple object tracking.* CVPR 2020

GNN-BASED ASSOCIATION

- A graph neural network (GNN) can be used to propagate node and edge embeddings over the graph

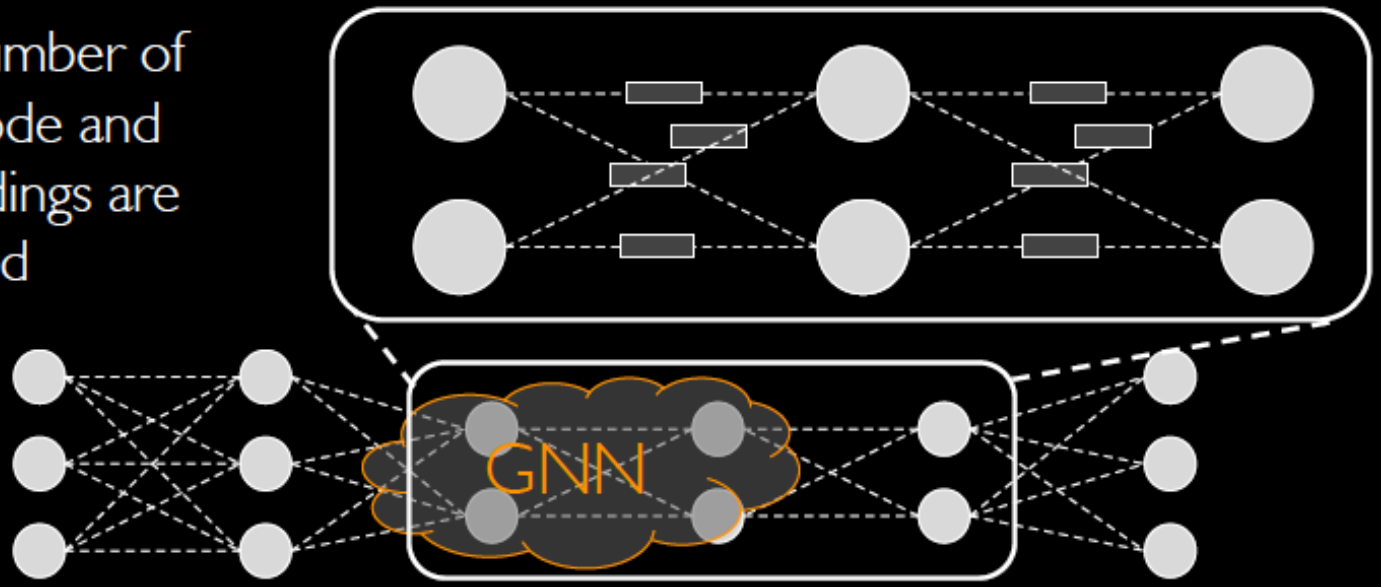
Input frames and object detections

Time

G. Brasó and L. Leal-Taixé. *Learning a neural solver for multiple object tracking.* CVPR 2020

GNN-BASED ASSOCIATION

For a fixed number of iterations, node and edge embeddings are updated



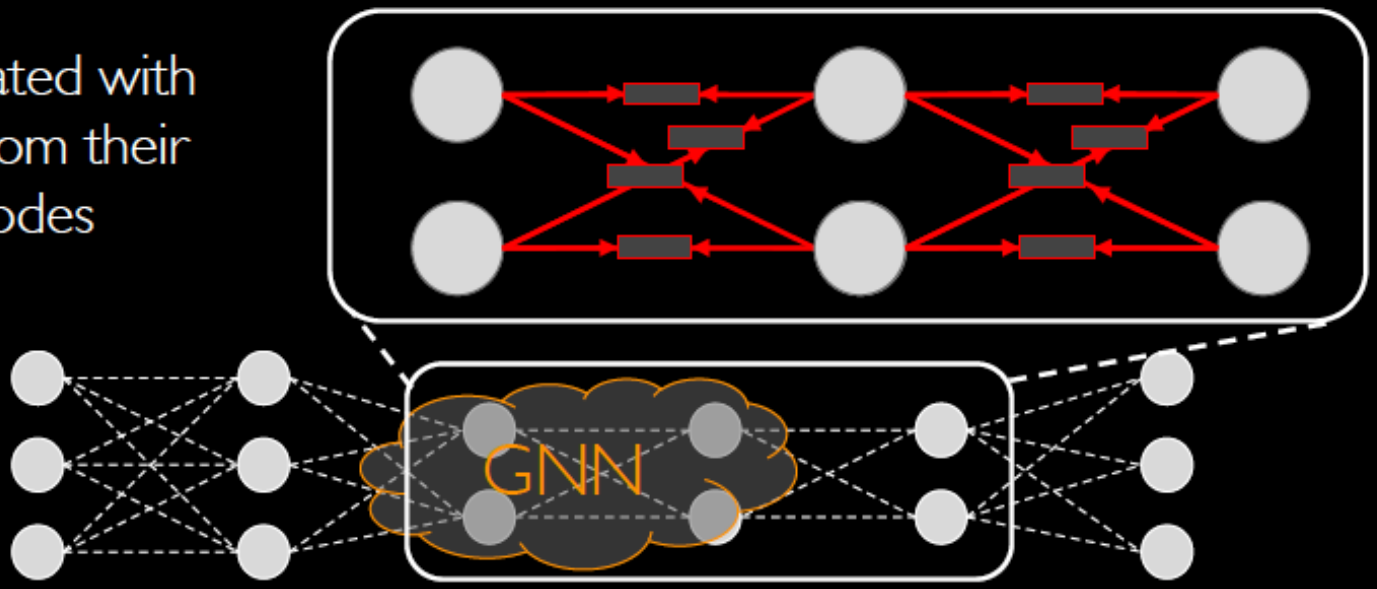
Input frames and object detections

Time

G. Brasó and L. Leal-Taixé. *Learning a neural solver for multiple object tracking.* CVPR 2020

GNN-BASED ASSOCIATION

Edges are updated with embeddings from their incident nodes



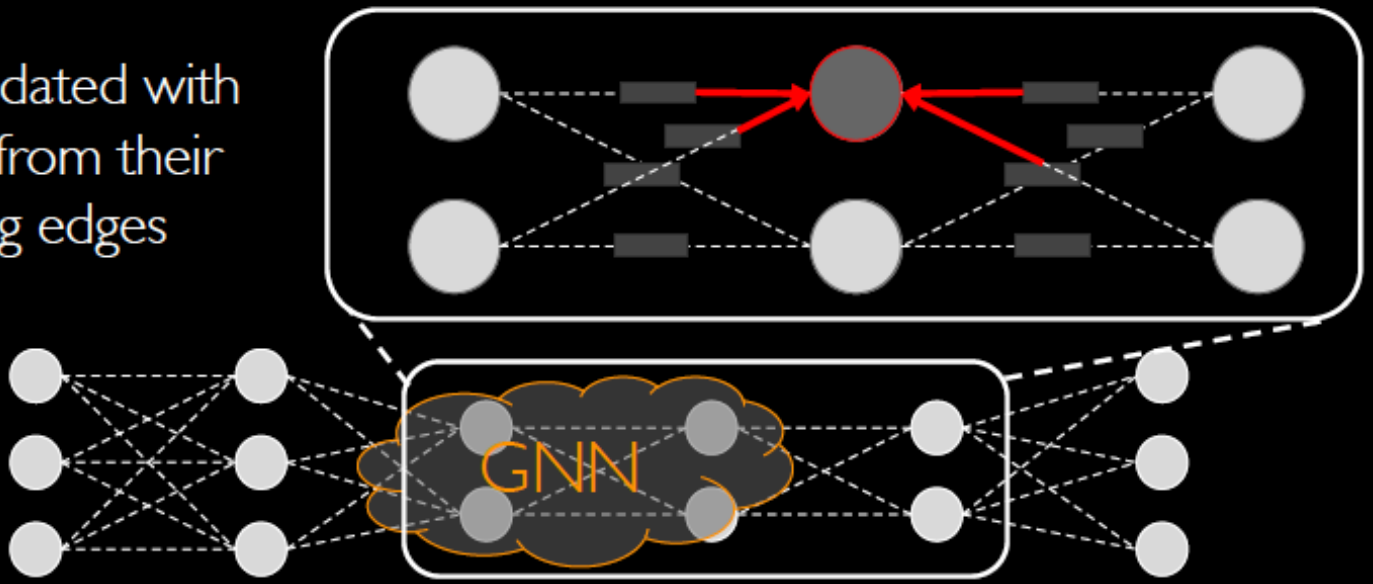
Input frames and object detections

Time

G. Brasó and L. Leal-Taixé. *Learning a neural solver for multiple object tracking.* CVPR 2020

GNN-BASED ASSOCIATION

Nodes are updated with embeddings from their neighboring edges



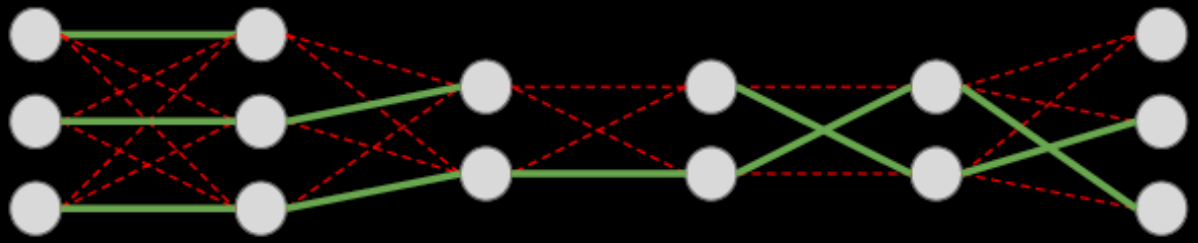
Input frames and object detections

Time

G. Brasó and L. Leal-Taixé. *Learning a neural solver for multiple object tracking.* CVPR 2020

GNN-BASED ASSOCIATION

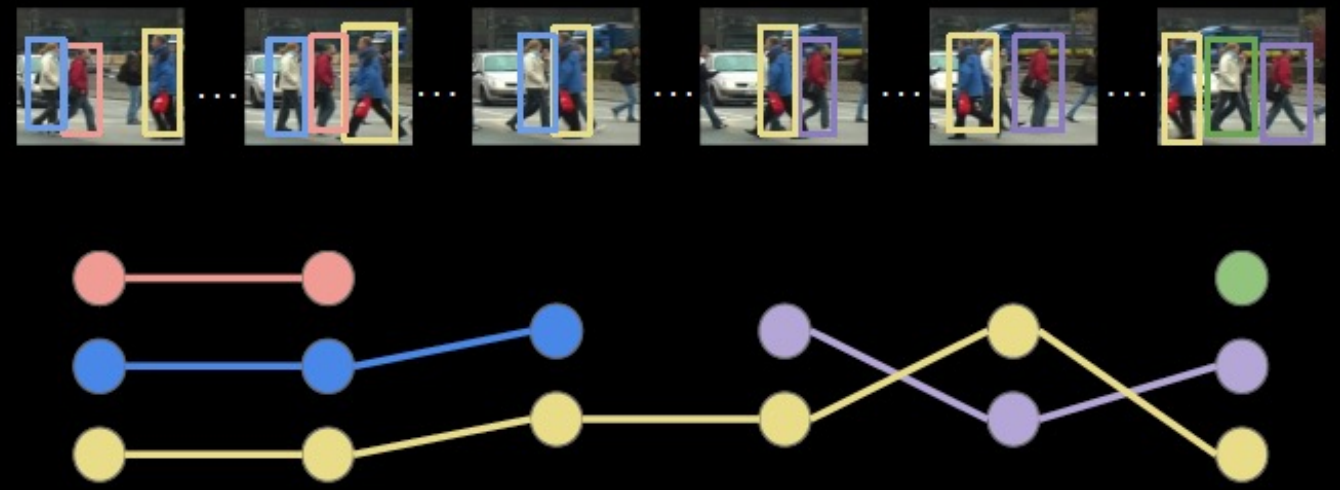
After neural message passing, edge embeddings are classified into correct and incorrect track hypotheses



Input frames and object detections

Time

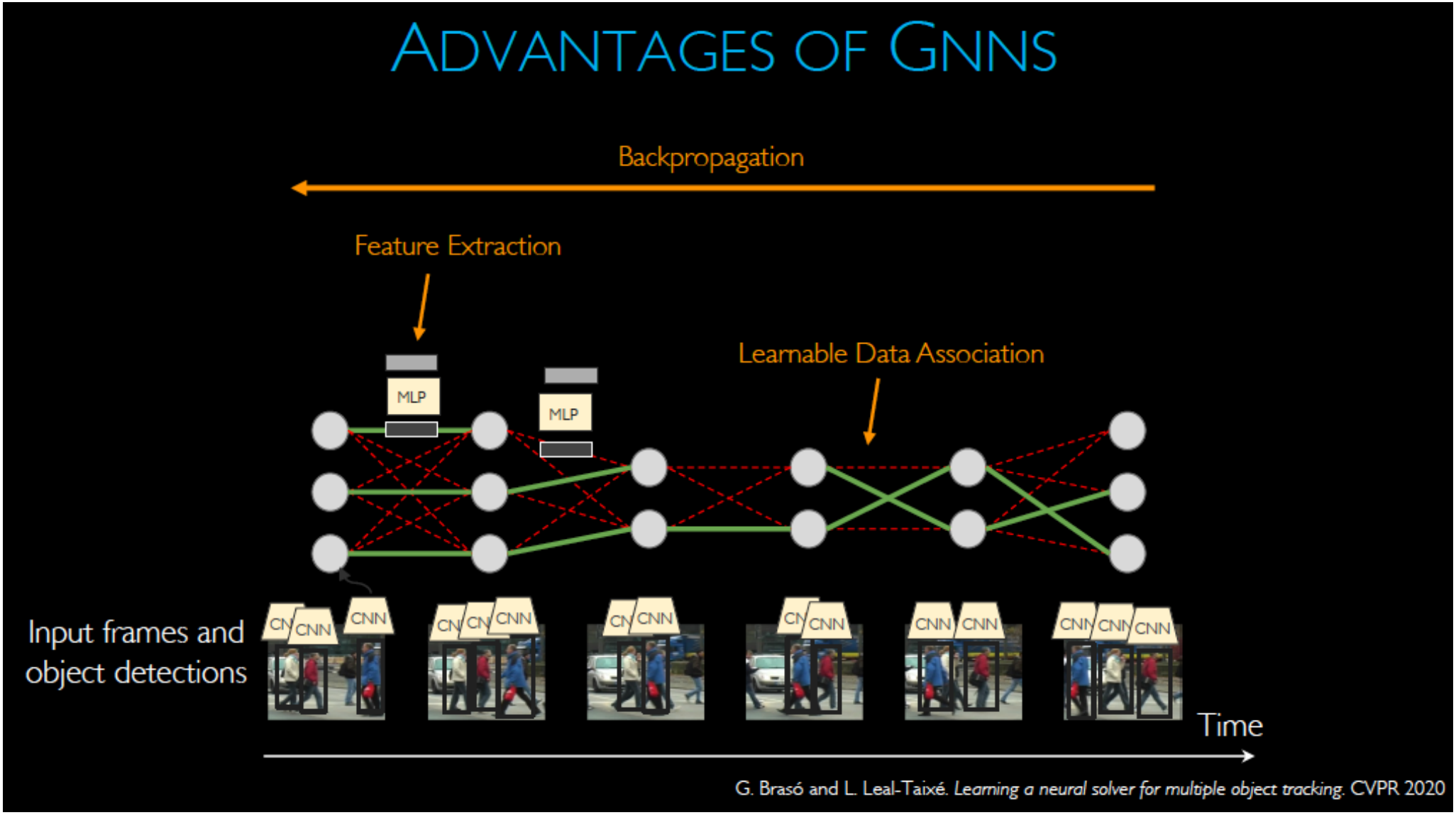
GNN-BASED ASSOCIATION



Input frames and object detections

Time

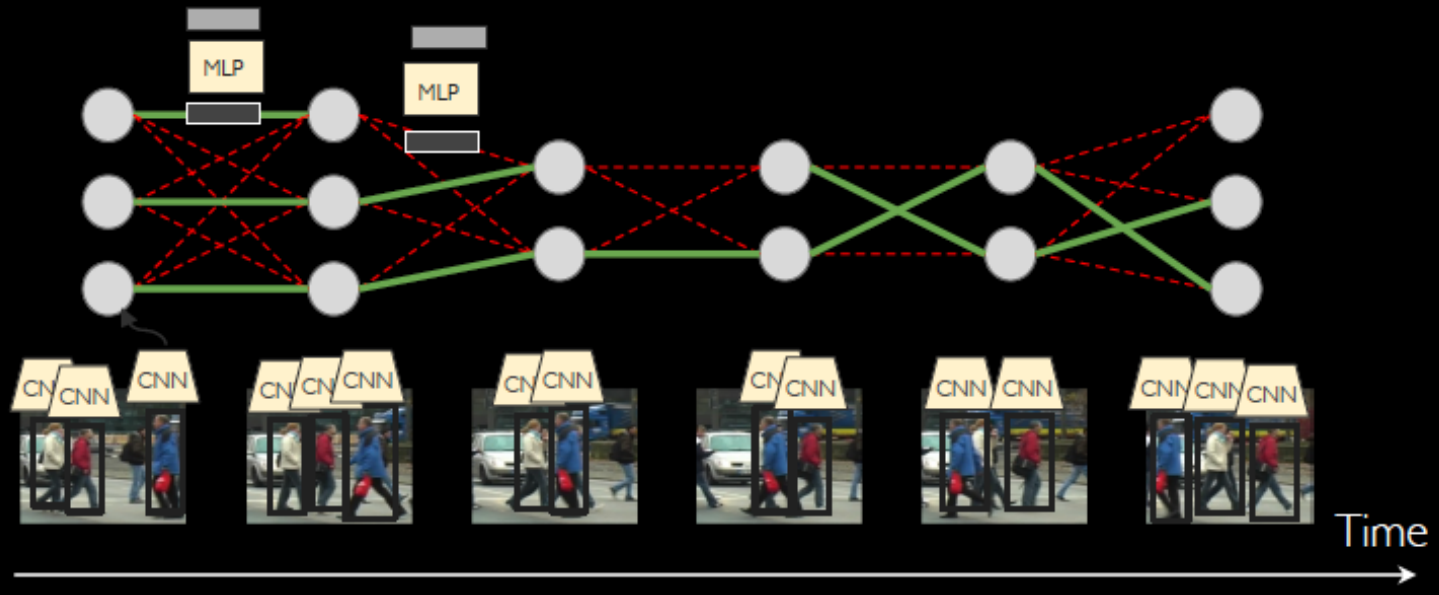
ADVANTAGES OF GNNS



G. Brasó and L. Leal-Taixé. *Learning a neural solver for multiple object tracking.* CVPR 2020

ADVANTAGES OF GNNS

- We can directly work in the **MOT domain** (graph)
- Learn features specifically for the task and the graph structure
- Avoid the need of expensive optimization at test time

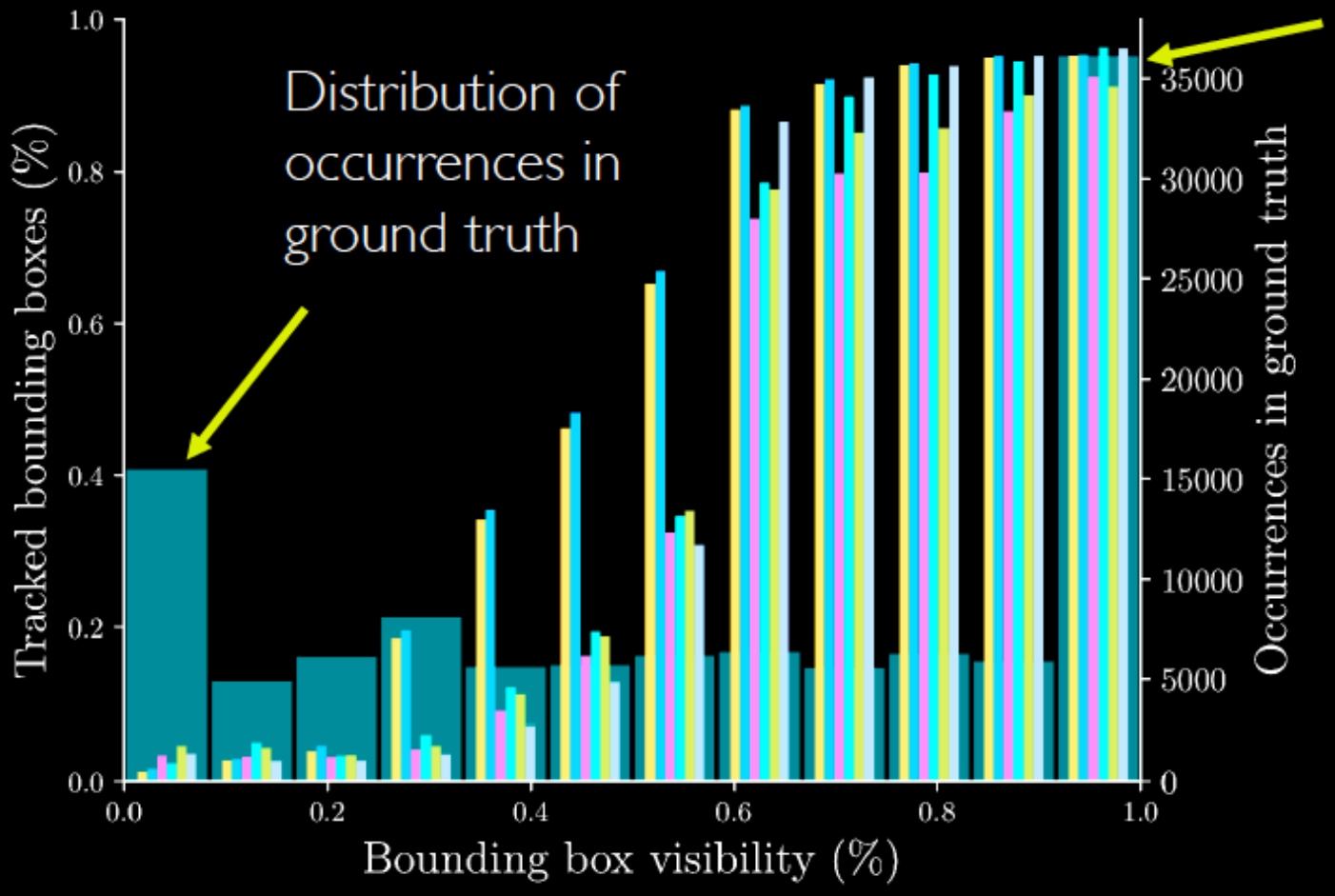


G. Brasó and L. Leal-Taixé. *Learning a neural solver for multiple object tracking.* CVPR 2020

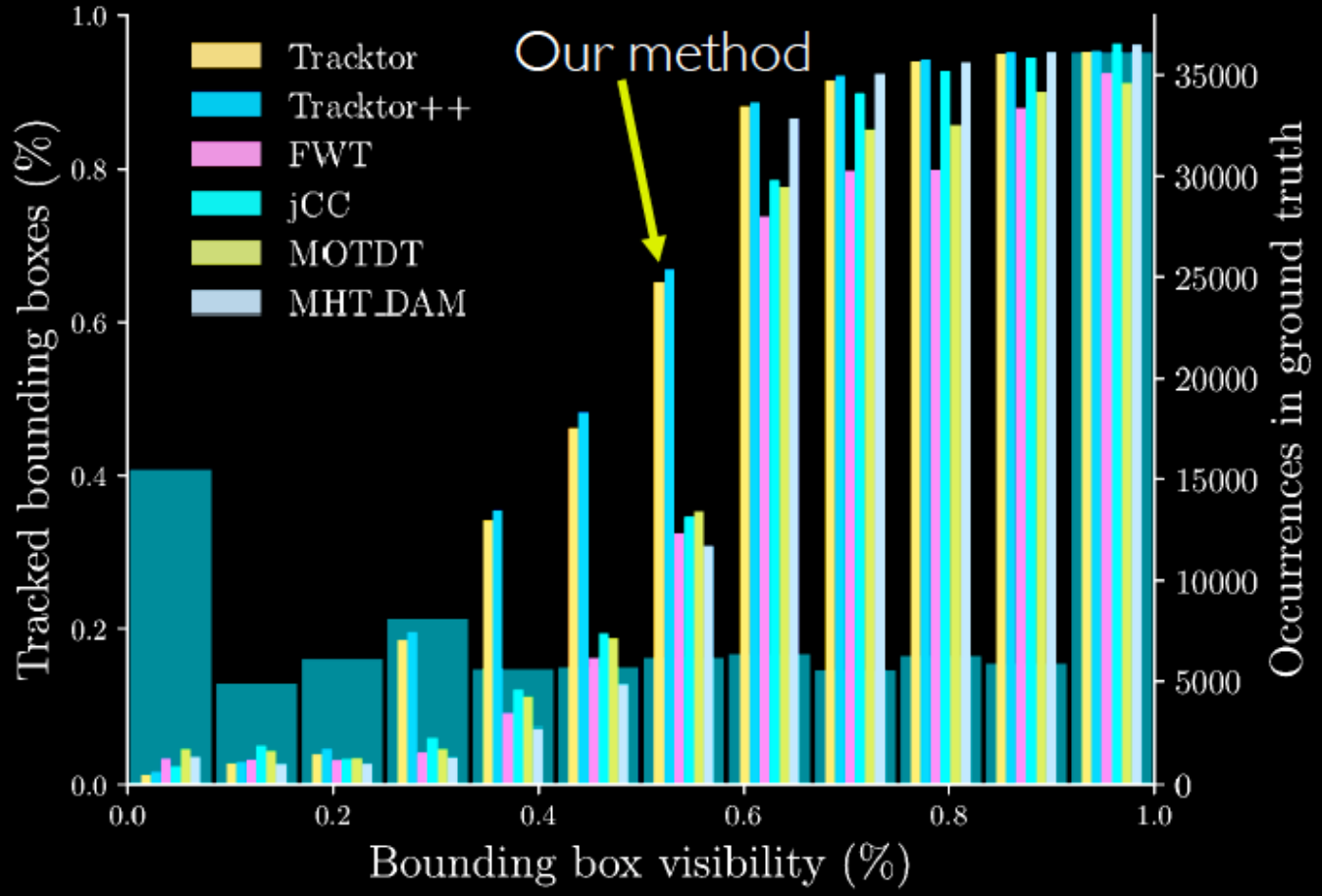
Major problems remain

Long term occlusions present problems

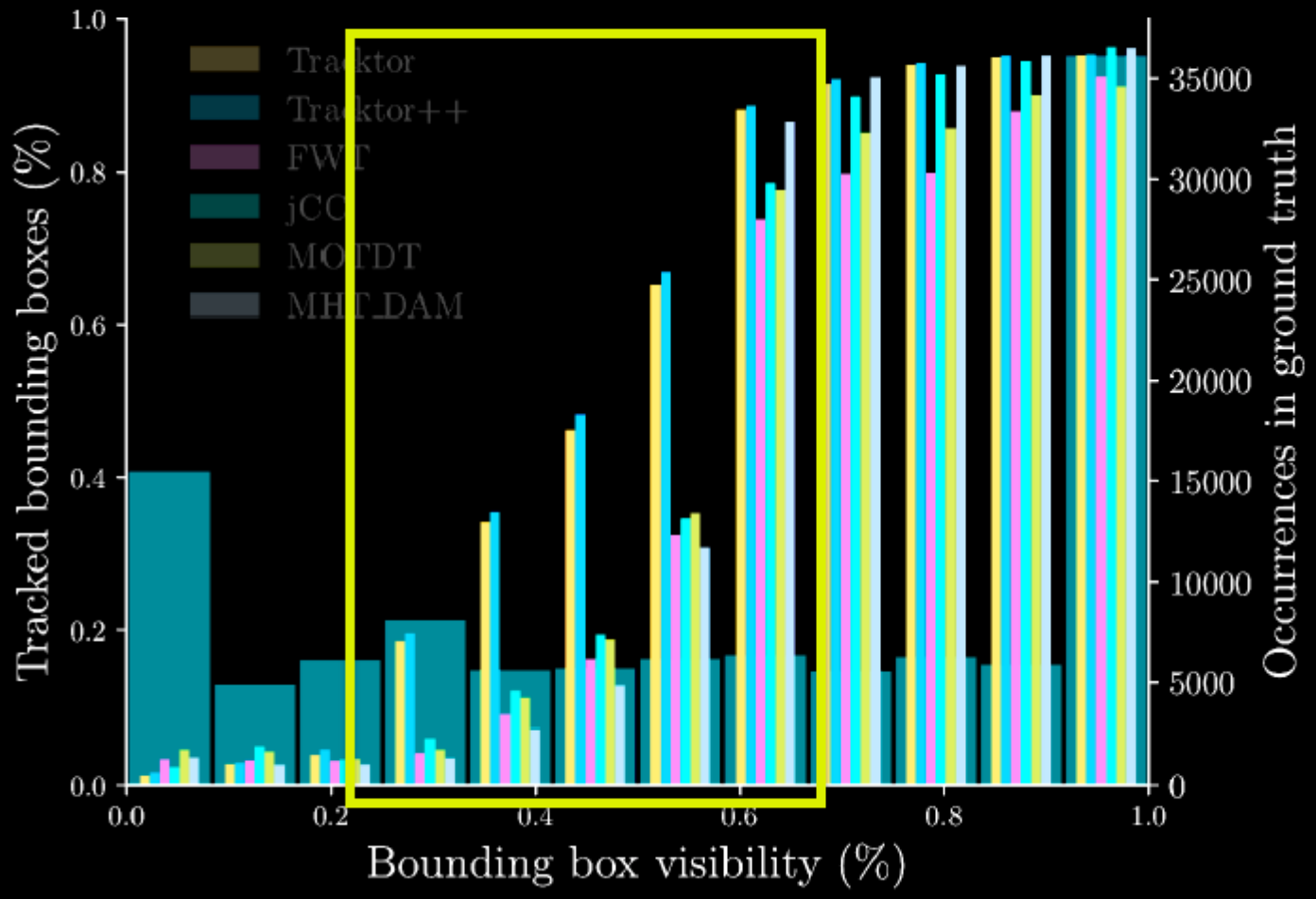
Analyzing the Results



Analyzing the Results



Analyzing the Results



None of the tracking-by-detection graph methods are more capable to recover from (partial) occlusions

Analyzing the Results

- Hard problems in tracking are left unsolved
  - coverage of large gaps in detections
  - recovering from partial occlusions
  - tracking of small targets

- All other methods are just marginally improving "easy" scenes

- In fact, accuracy in tracking has only increased by 2.4 percentage points between 2017 and 2019 for MOT16 in MOTChallenge

TRACKING-BY-ATTENTION*

*Attention jointly solves the detection and tracking task.

Quick and (very) dirty guide to transformers - I

Mapping a sequence to a sequence is an important problem

eg machine translation

seq. of Latin words -> seq. of English words

but words affect other words, and are mangled as to order

Gallia est omnis divisa in partes tres; quarum unam incolunt Belgae,

All Gaul is divided into three parts; of which the Belgians inhabit one, the Aquitanians

aliam Aquitani, tertiam qui ipsorum linguā Celtae, nostrā Galli,

(inhabit) another (part), (those) who are called Celts in their language, in ours Gauls,

appellantur. Hi omnes linguā, institutīs, legibus inter se differunt.

(inhabit) the third (part). All these differ amongst themselves in language, customs, (and) laws.

Quick and (very) dirty guide to transformers - II

Idea:

build architecture ensuring rep'n of pred word

is affected by long range of inputs

attention

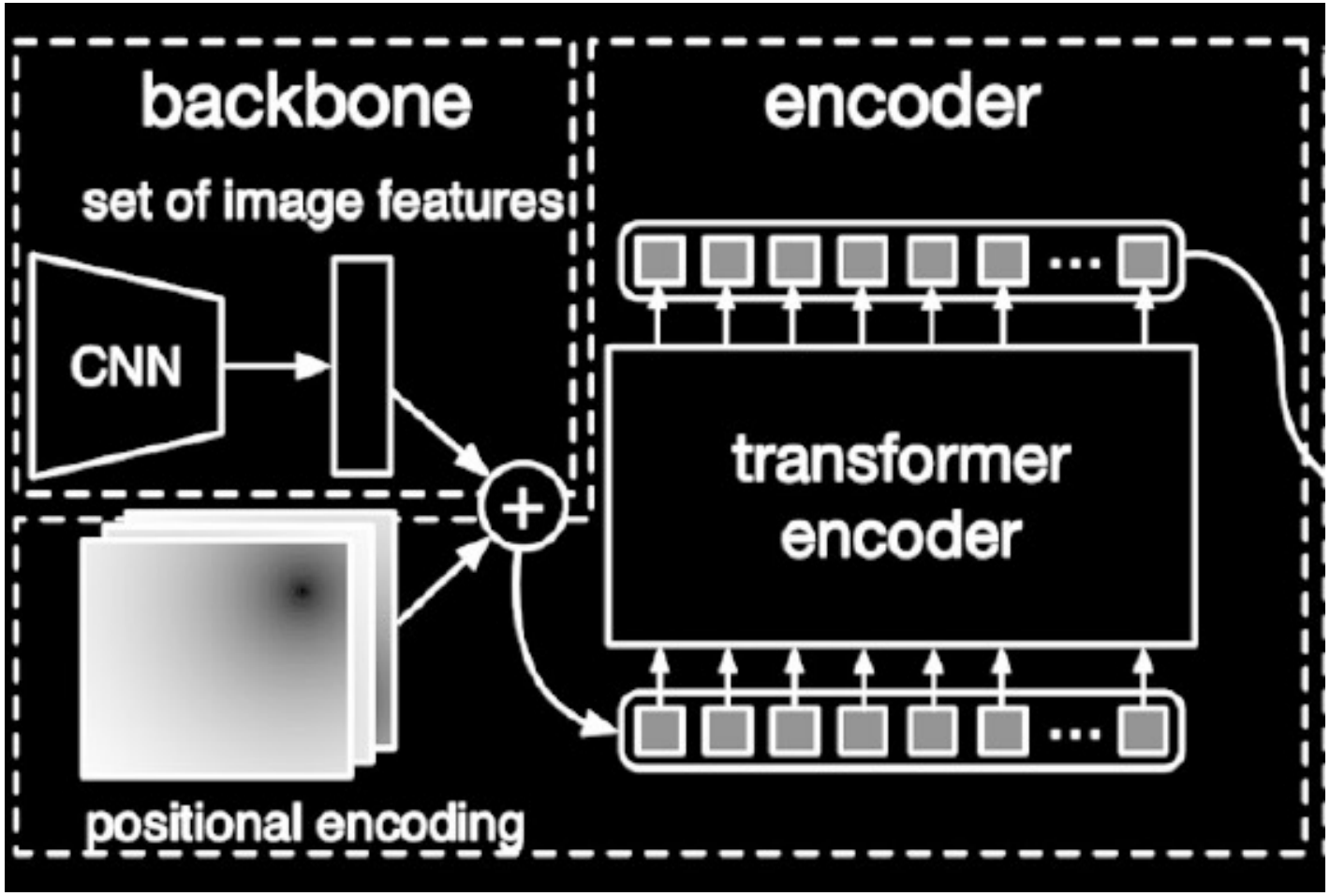
word rep'n is weighted average of other word rep'ns

weights depend on "similarity" score

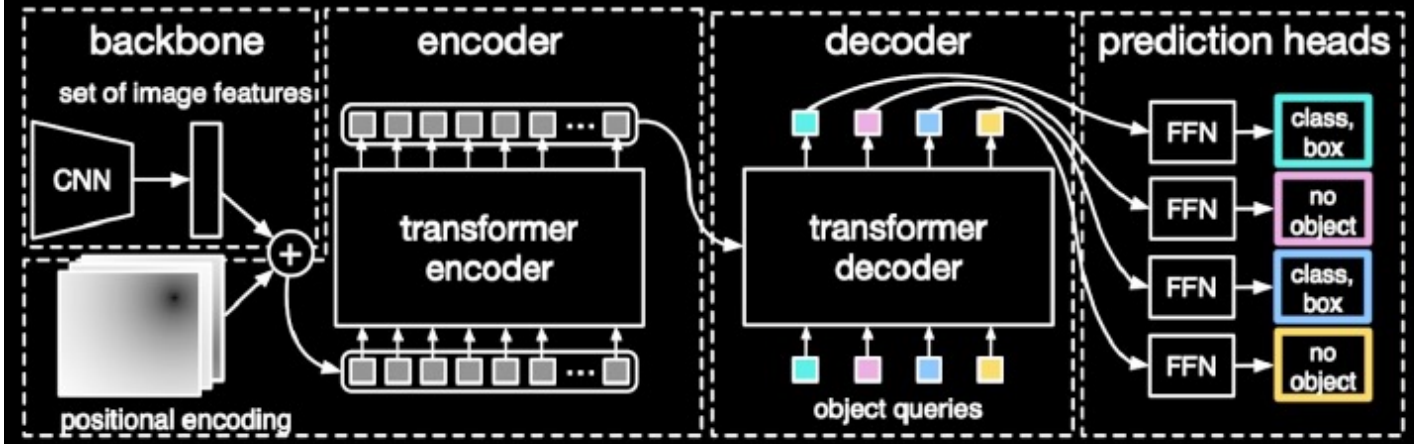
Outcome:

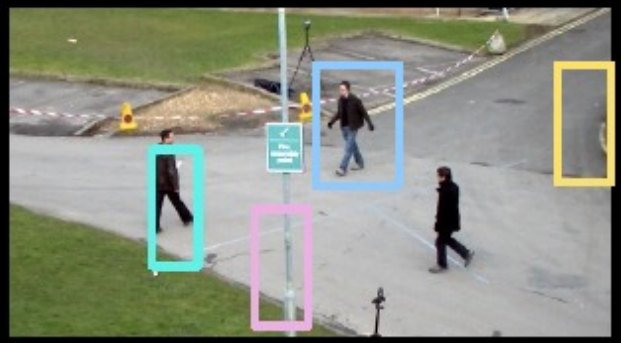
revolutionary improvements in machine translation accuracies

But images aren’t sequences…



DETECTION WITH TRANSFORMERS



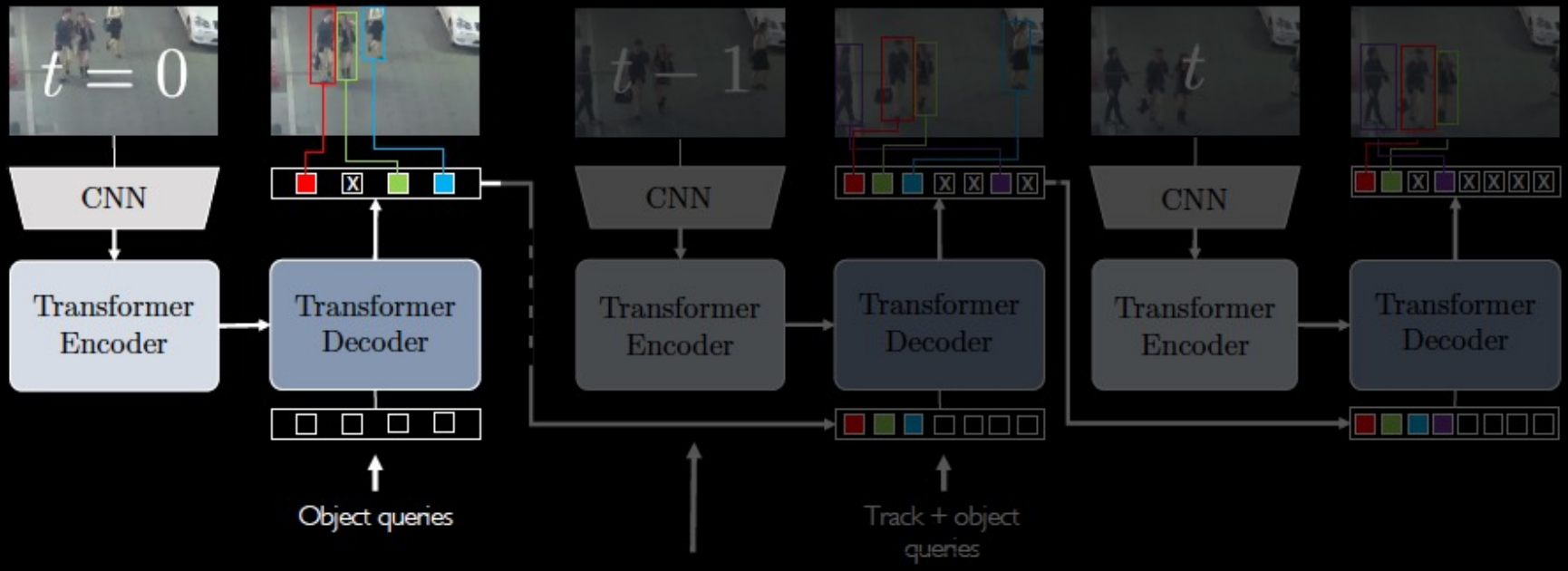


- Object detection a set prediction problem [1, 2]
- Transformer decoder
  - Object query self-attention ({class, box} or no-object)
  - Encoded image feature and object query cross attention

[1] Carion et al. *End-to-End Object Detection with Transformers*. ECCV, 2020.
[2] Zhu et al. *Deformable DETR: Deformable transformers for end-to-end object detection*. ICLR 2021.

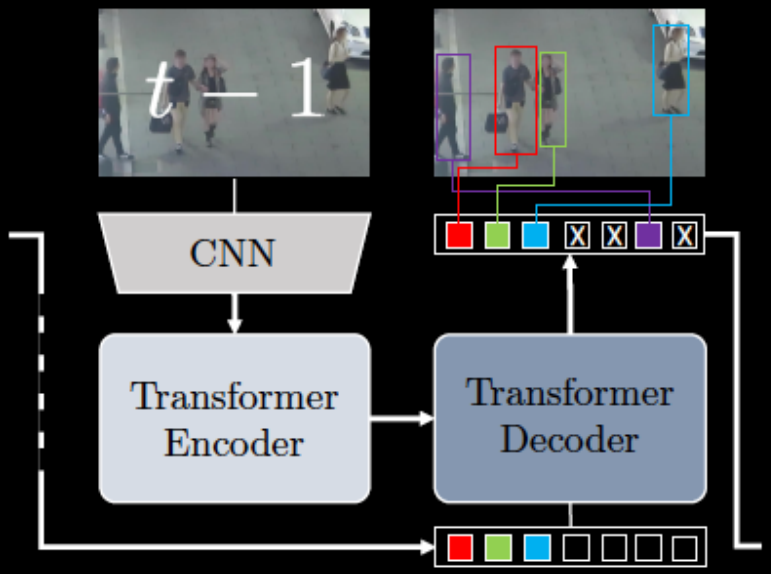
TRACKFORMER

- MOT as a frame-to-frame set prediction problem



Meinhardt et al. *Trackformer: multi-object tracking with transformers.* CVPR 2022

Transformer Query Decoding



1. Self-attention between queries
   a. Initialize new track (object query)
   b. Terminate occluded track

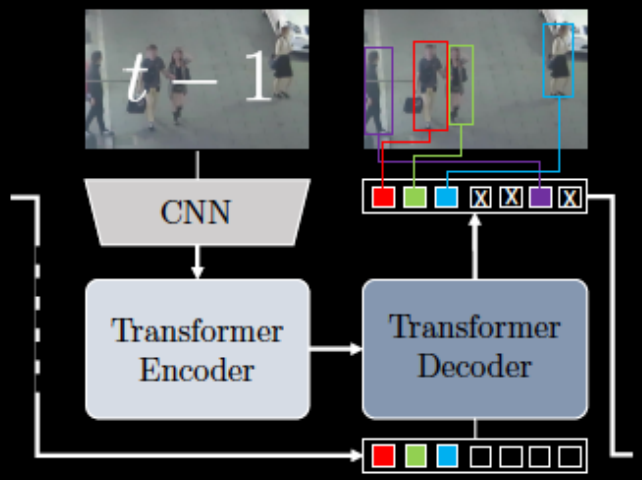
2. Encoder-decoder attention
   a. Find new object in frame
   b. Adjust to changed position of tracks

Can I Recover From Occlusions?

👍 Just keep track queries active for a time window.

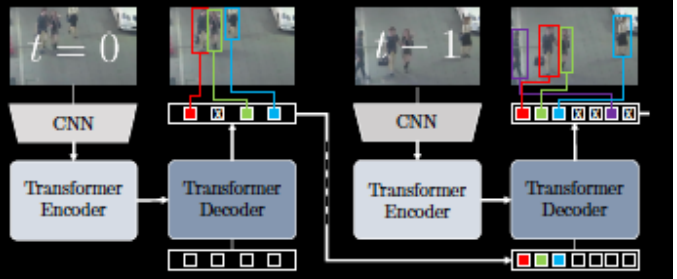
👍 No need to an extra re-ID head.

👎 The spatial information embedded into each track query prevents their application for long-term occlusions.



TRAINING

2 frames

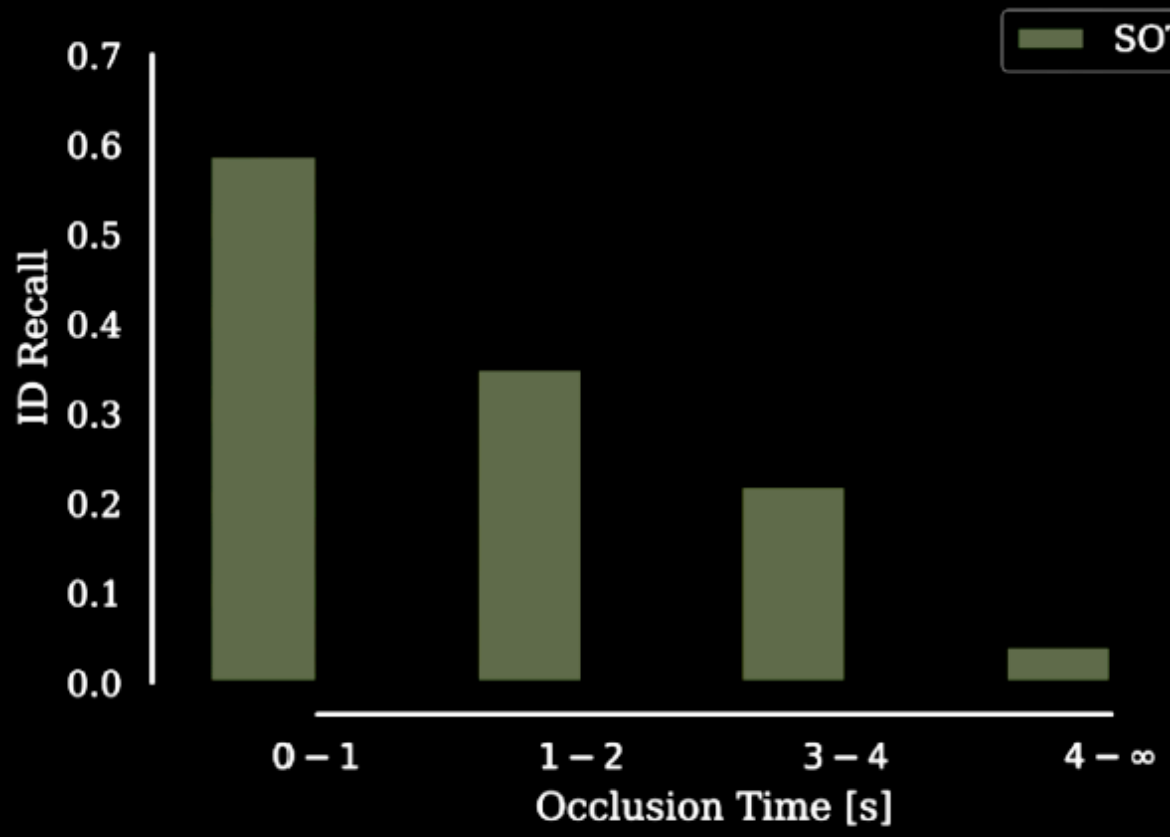


1. Object detection on frame $t - 1$ with N_{object} object queries
2. Tracking of objects from (1.) and detection of new objects on frame t with all $N = N_{\text{object}} + N_{\text{track}}$ queries
3. Assign N predictions to ground truth objects in t
4. Compute set-prediction loss:
   a. Classification (pedestrian or no-object)
   b. Bounding box

TrackFormer

- Elegant formulation of tracking which naturally merges detection and data association
- Good performance with partial occlusions
- Good performance where detectors are weak
- State-of-the-art results (with some data and some tricks)
- Similar concurrent papers: MeMOT (ECCV22) and MOTR (CVPR22)

Research into Long Occlusions



Research into Long Occlusions

- If we have long gaps in a trajectory without detection, there are 2 things we can do:
  - Re-identification [1]
  - Trajectory prediction – inherently in 3D

[1] Seidenschwarz et al. *Simple cues lead to a strong multi-object tracker*. arxiv 2022.

What's Next for MOT?

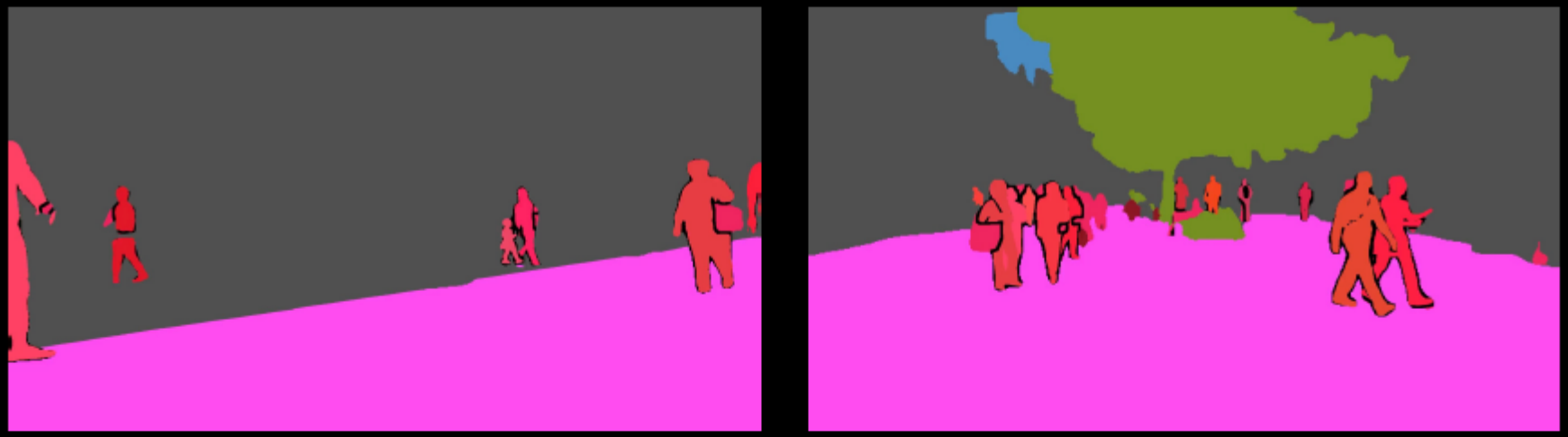
Obtaining Data

- Synthetic data: a study on how synthetic data can help us train detectors, trackers, and re-identification methods.

Fabbri et al. *MOTSynth: How Can Synthetic Data Help Pedestrian Detection and Tracking?*. ICCV 2021.

Increasing Accuracy

- STEP: Segmenting and Tracking Every Pixel

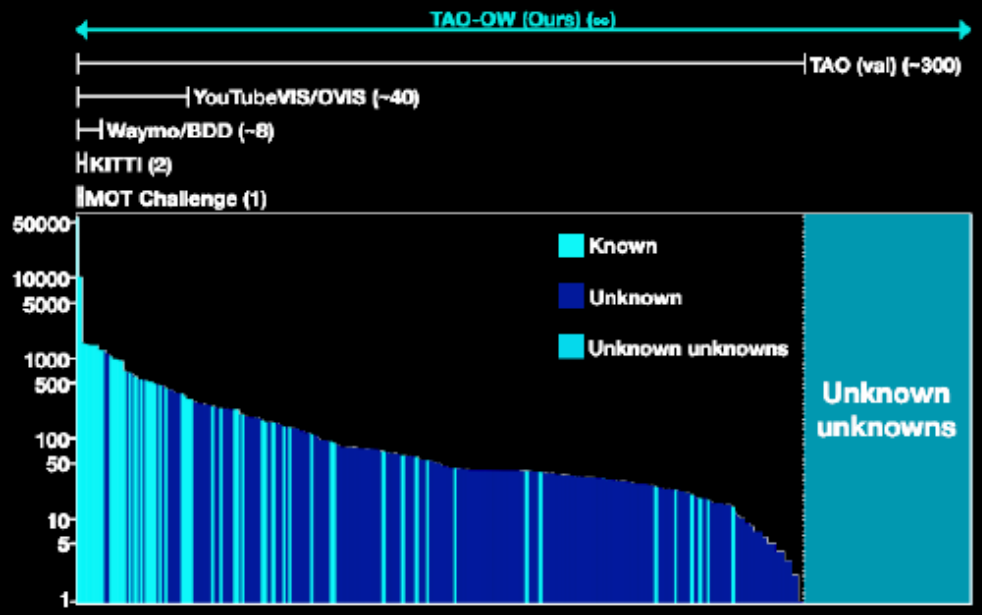


Weber et al. *STEP: Segmenting and Tracking Every Pixel.* NeurIPS Track Datasets and Benchmarks 2021.

Open World

- Bringing video understanding to the open world – for classes for which we do not have pre-trained detectors

- Open-world tracking



Liu et al. *Opening up open-world tracking.* CVPR 2022.