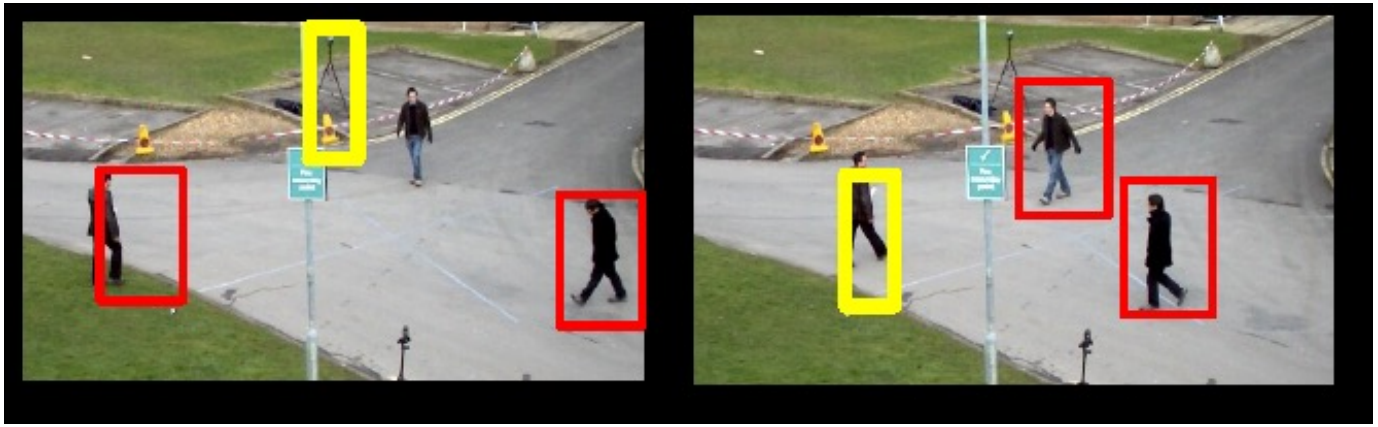
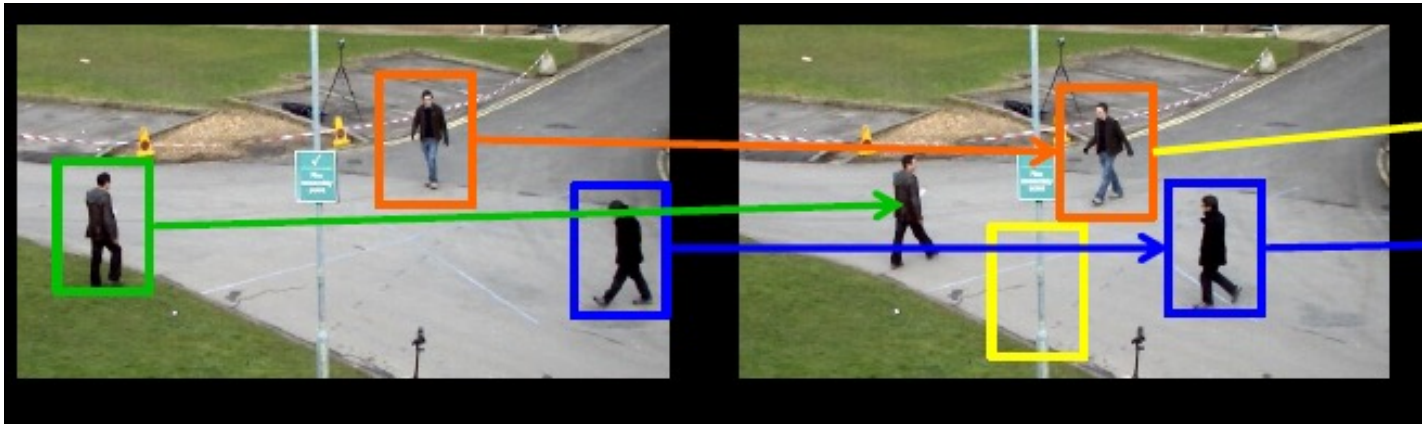
Tracking by Detection

D.A. Forsyth

University of Illinois at Urbana-Champaign

Multi-object tracking



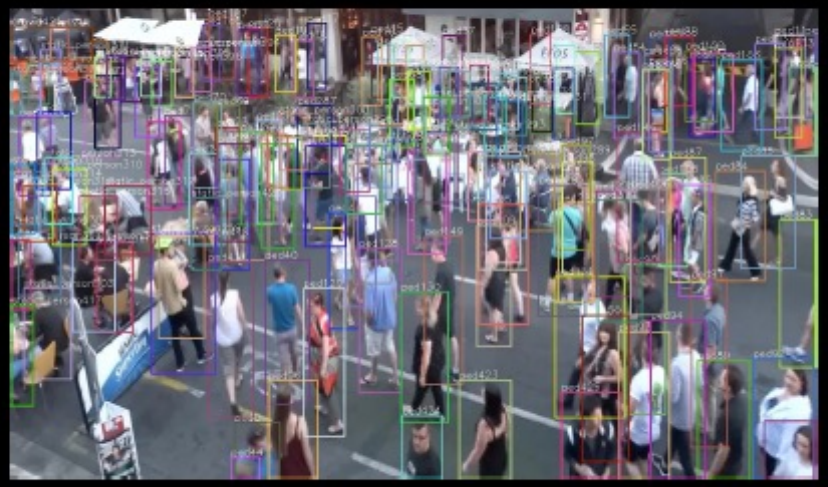


Tracking – Why?

- 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.)

CHALLENGES

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



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

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

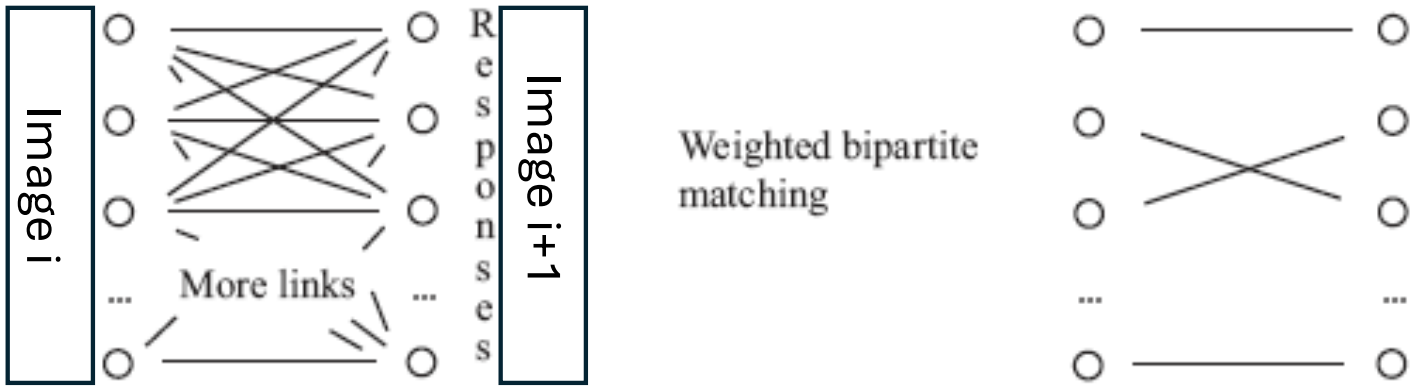
OFFLINE VS. ONLINE

- 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

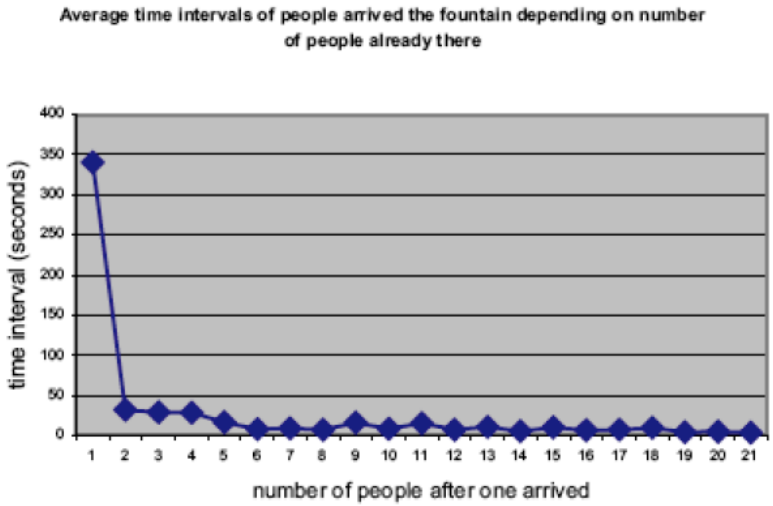
- Offline tracking
  - Processes a batch of frames
  - Good to recover from occlusions (short ones as we will see)
  - Not suitable for real-time applications
  - Suitable for video analysis, automatic labeling, video editing.

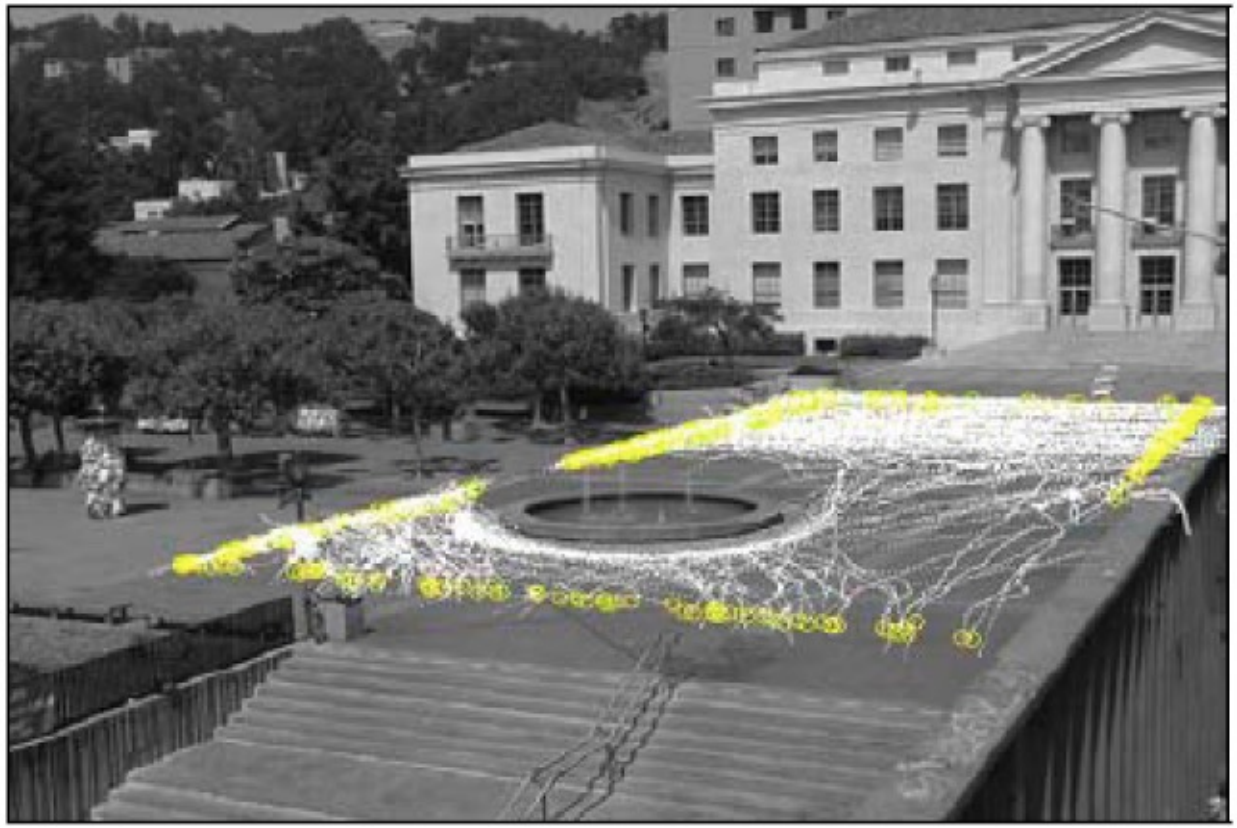
Tracking by detection

- Assume
  - a reliable detector
  - detections that are well spaced in images
    - (or have distinctive properties)
    - e.g. news anchors; heads in public
- Link detects across time
  - weighted bipartite matching



Point tracks reveal public behaviour

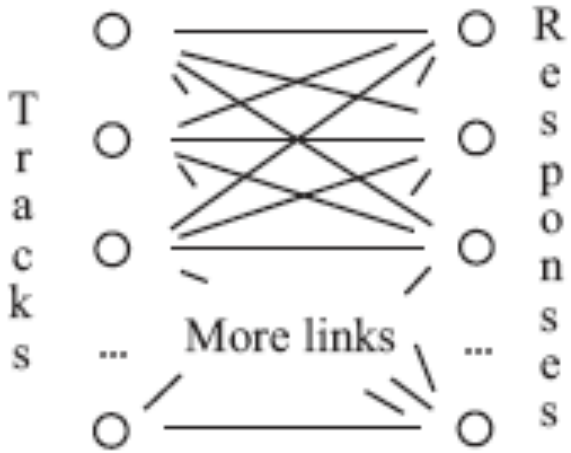




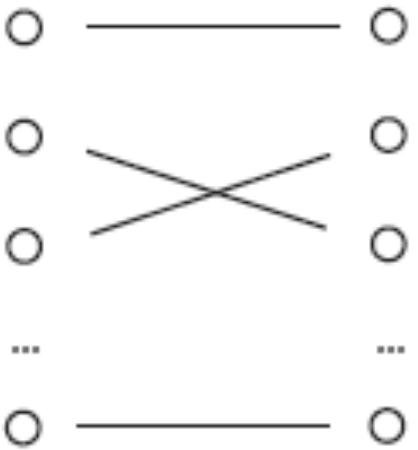
Yan+Forsyth, 04

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



Weighted bipartite matching



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.

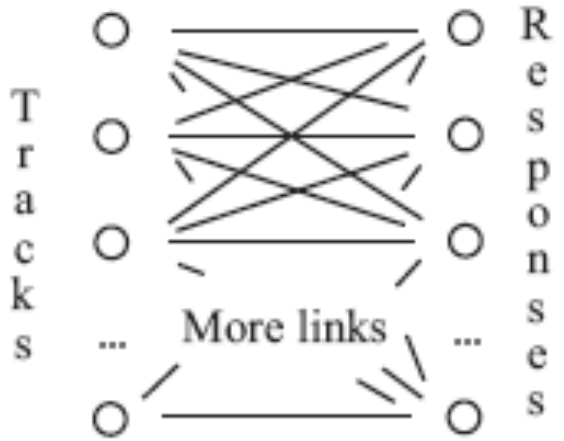
First frame: Create a track for each detector response.

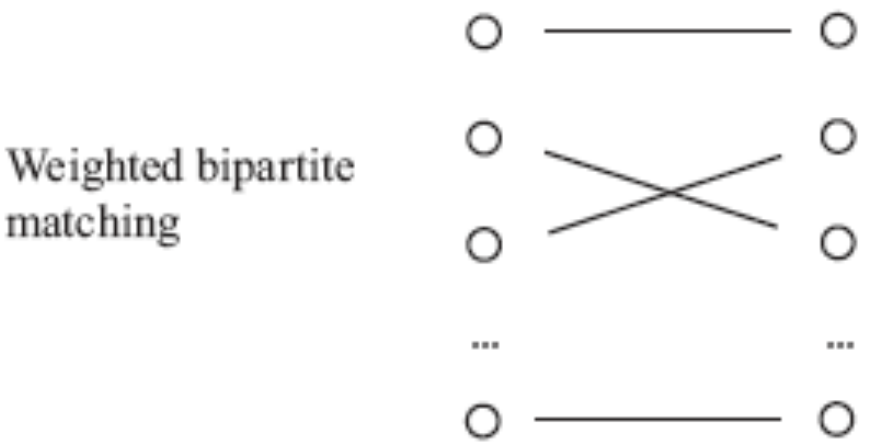
N'th frame:
Link tracks and detector responses by solving a bipartite matching problem.
Spawn a new track for each detector response not allocated to a track.
Reap any track that has not received a detector response for some number of frames.

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.

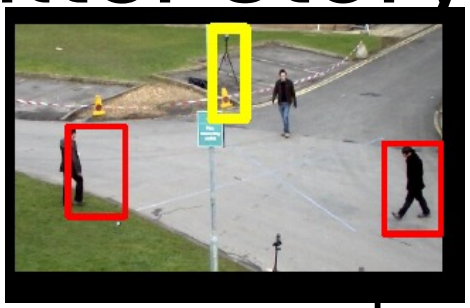
TBD (to date)





- Questions:
- can we use dynamics to simplify matching?
- how do we represent similarity?
-

Recall Kalman Filter story



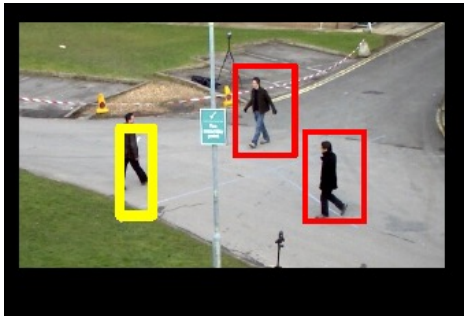
Have:

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

Attaching a Kalman filter

- Use filter to represent, update tracks
  - one filter to manage state for each track
- Associate measurements to tracks with WBM
- Now update filter
  - usually, a constant velocity model is enough
  - sometimes, velocity is zero and noise is large

A Simple Online Tracker

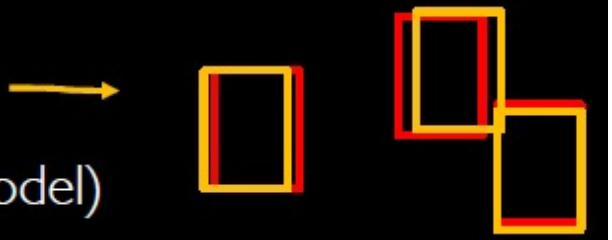
t

t+1

t+2

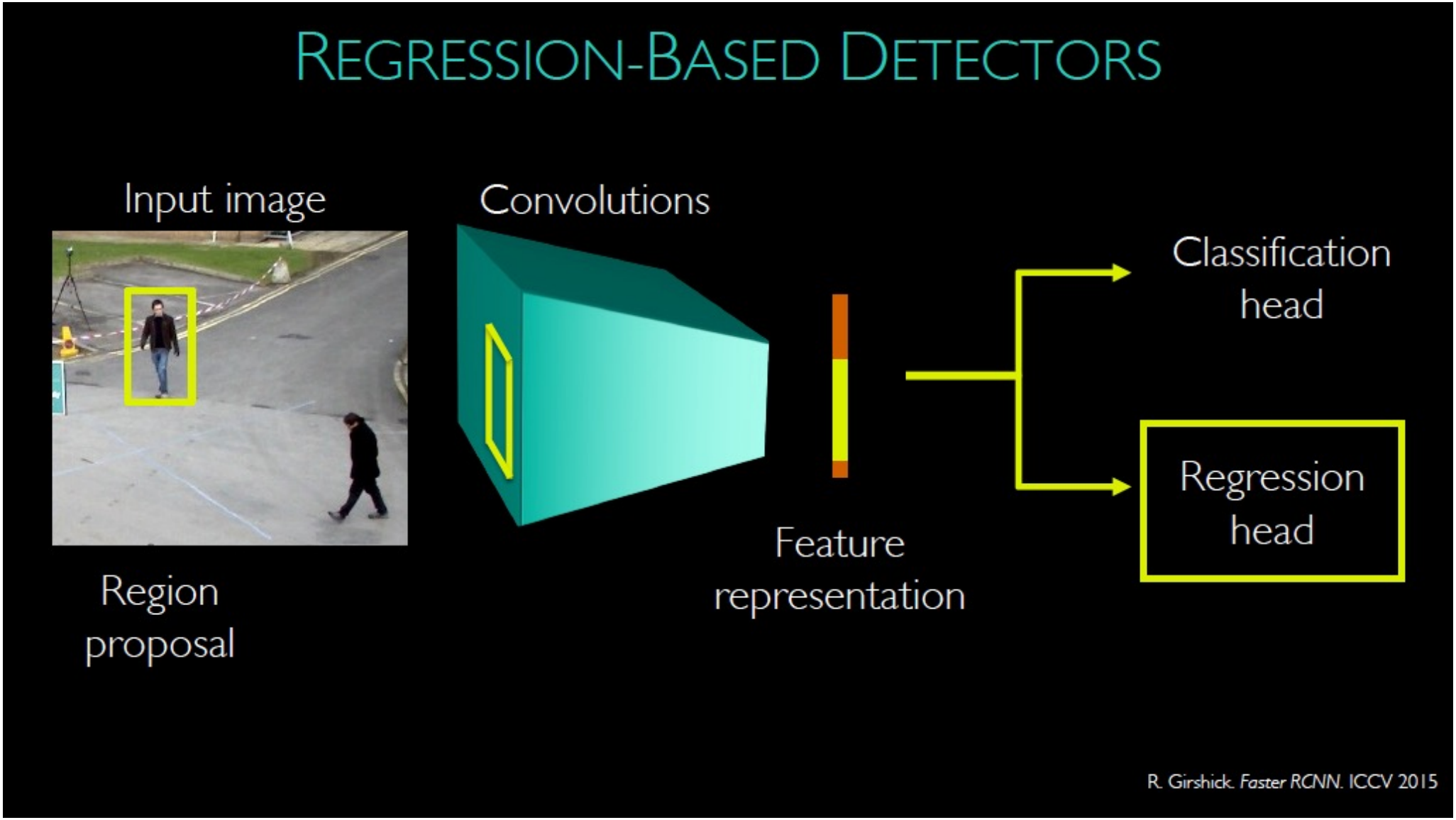
- 1. Track initialization (e.g. using a detector)

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

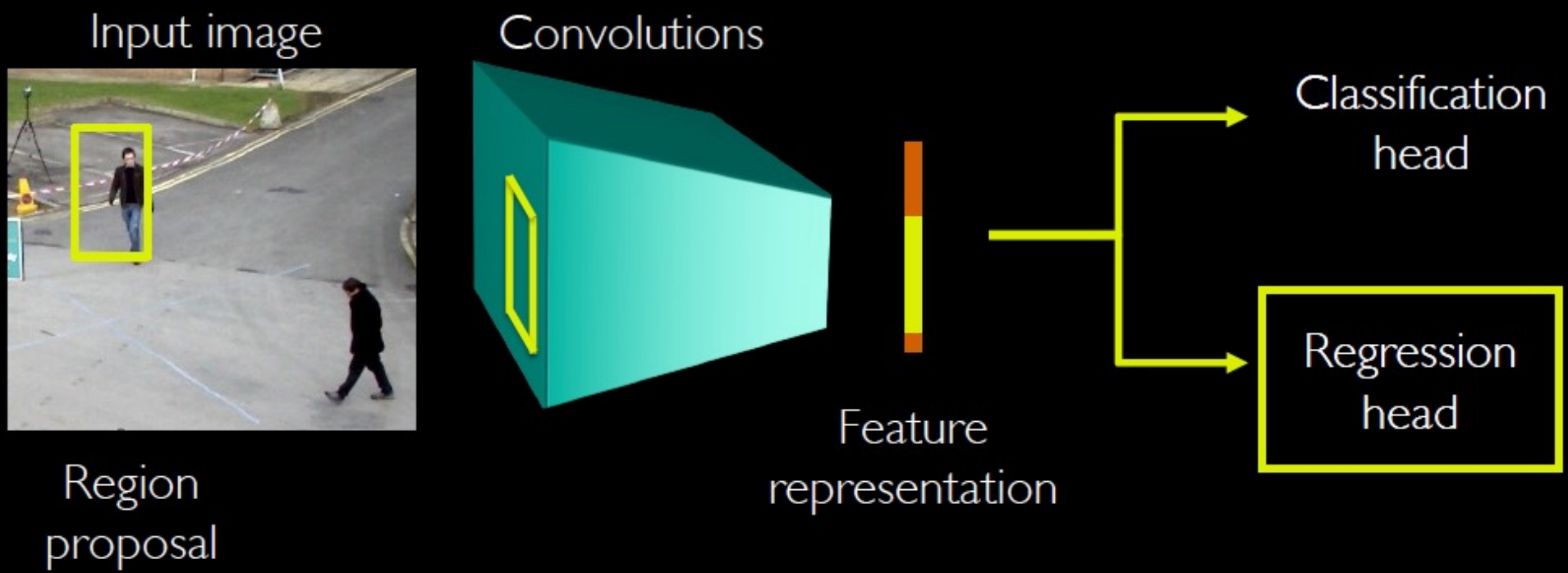


- 3. **Matching** predictions with detections (appearance model)

Recall Faster RCNN

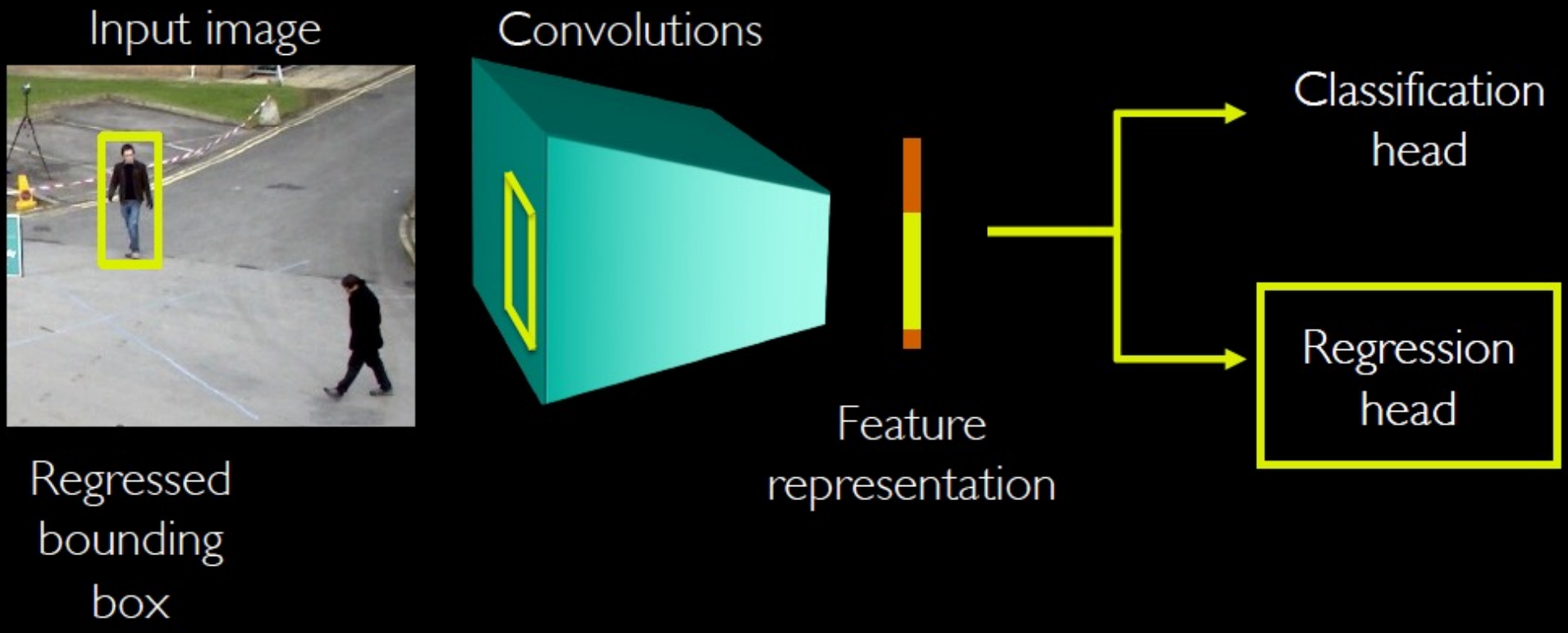


Regression-Based Detectors



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

- 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.

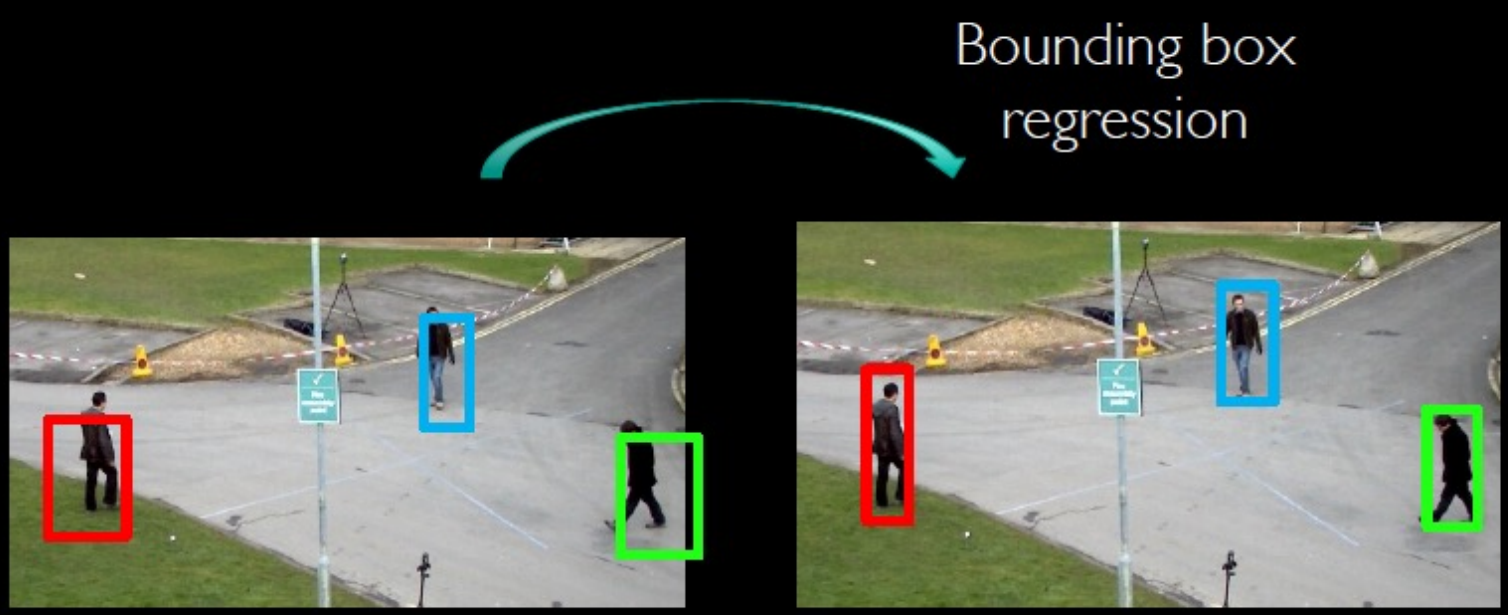
From Detector to Tracktor

Frame t+1

Use detections of frame t as proposals

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

From Detector to Tracktor



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