Self-Driving Cars

Lecture 11 – Object Tracking

Prof. Dr.-Ing. Andreas Geiger

Autonomous Vision Group

University of Tübingen / MPI-IS

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

ellis
European Laboratory for Learning and Intelligent Systems

Agenda

11.1 Introduction

11.2 Filtering

11.3 Association

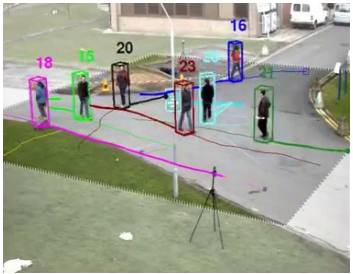
11.4 Holistic Scene Understanding

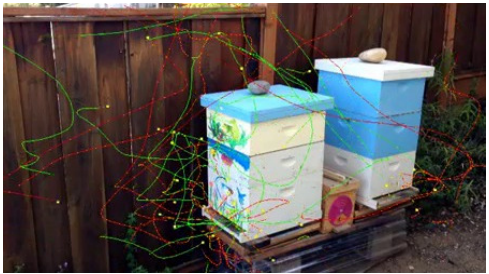
11.1
Introduction

What is Tracking?

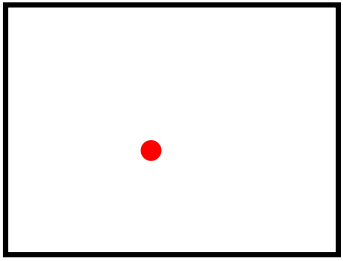
Goal:

- Given noisy object detections (e.g., bounding boxes) for each frame of a sequence, **associate** those that belong to the same physical object
- Reject detections that are false alarms; **initiate/delete** object tracks
- Estimate **object state** by considering an observation/motion model

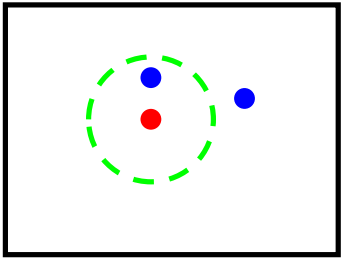




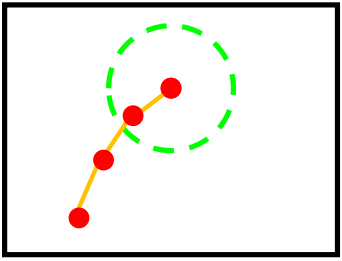
Elements of Tracking



Detection



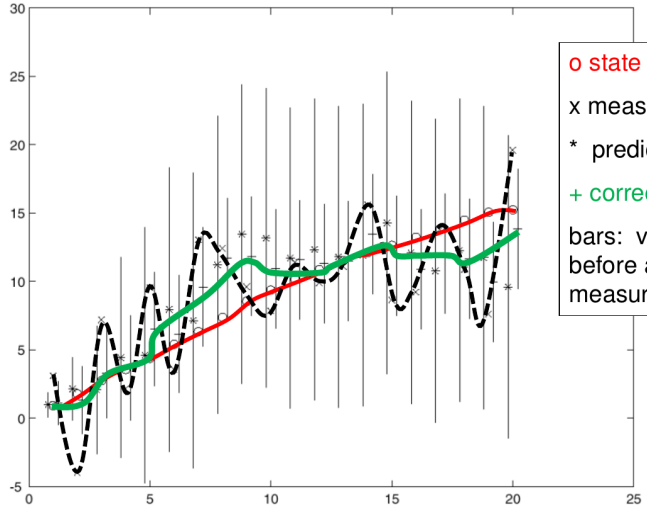
Association



Filtering

- **Detection:** Where are candidate objects in each frame? (“tracking-by-detection”)
- **Association:** Which detection corresponds to which object?
- **Filtering:** What is the most likely object state, e.g., location and size?
  (Detections are noisy ⇒ exploit probabilistic observation/motion models)

Filtering



o state

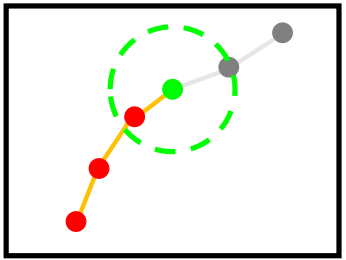
x measurement

* predicted mean estimate

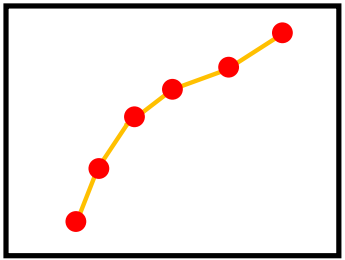
\+ corrected mean estimate

bars: variance estimates before and after measurements

Online vs. Offline Tracking



Online Tracking



Offline Tracking

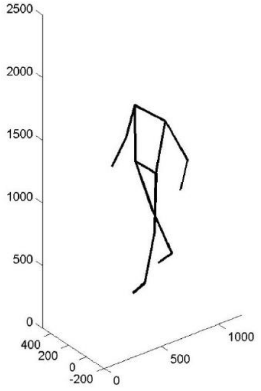
- **Online Tracking:** Estimate current state given current and past observations
- **Offline Tracking:** Estimate all states given all observations (batch mode)
- As we consider self-driving, we will focus on online tracking in this lecture

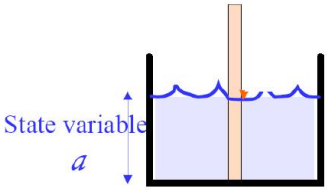
Slide Credits / Further Readings

- Bastian Leibe (RWTH Aachen): Computer Vision 2
  `https://www.vision.rwth-aachen.de/course/25/`
- Laura Leal-Taixé (TUM): Computer Vision 3: Detection, Segmentation, Tracking
  `https://dvl.in.tum.de/teaching/cv3dst-ss20/`

11.2
Filtering

State vs. Observation

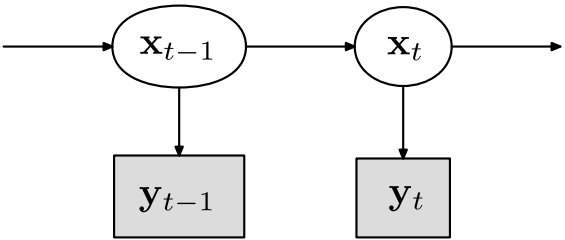




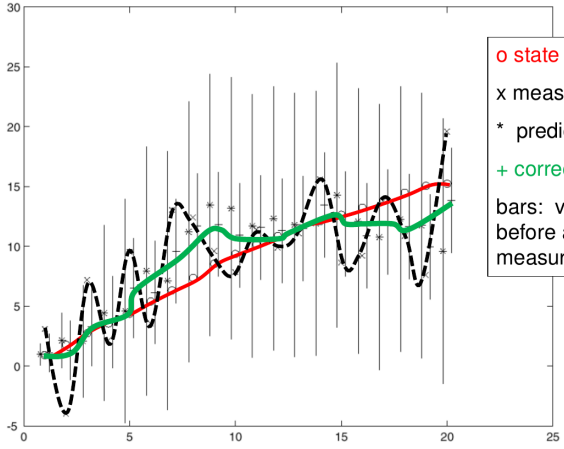
- **Hidden state:** parameters of interest (e.g., object location)
- **Observation:** what we directly observe (=measurement)
- Due to sensor noise/observability: hidden state $\neq$ observation

Representation

- For now, let's assume that we have only a **single object** which we want to track
- The moving object of interest is characterized by an underlying state $\mathbf{x}$
- State $\mathbf{x}$ gives rise to measurements or observations $\mathbf{y}$
- At each time t, the state changes to $\mathbf{x}_t$ and we receive a new observation $\mathbf{y}_t$
- Given a sequence of observations $\mathbf{Y} = \{\mathbf{y}_t\}$, we aim to recover $\mathbf{X} = \{\mathbf{x}_t\}$



Filtering



o state

x measurement

* predicted mean estimate

+ corrected mean estimate

bars: variance estimates before and after measurements

- In tracking, we often aim for a **probabilistic estimate,** not a point estimate

Probability Theory Recap

Probability Theory Recap

Random Variables:

- Discrete random variable: $x \in \{1, \dots, C\}$
  - Probability that x takes value c: $p(x = c)$
- Continuous random variable: $x \in \mathbb{R}$
  - Probability that x takes value in $\mathbb{A} \subset \mathbb{R}$: $p(x \in \mathbb{A})$
- Distribution over x: $p(x)$ as short notation for $p(x = c)$
- Joint distribution: $p(x, y)$ as short notation for $p(x = c, y = c')$

Properties:

- Marginal distribution: $p(x) = \sum_y p(x, y)$ or $p(x) = \int_y p(x, y)$
- Product rule: $p(x, y) = p(x|y)p(y)$

Bayes' Rule

Based on the product rule $p(x, y) = p(x|y)p(y)$ we have:

$$p(x, y) = p(x|y)p(y) = p(y|x)p(x) = p(y, x)$$

Solving for $p(x|y)$ yields **Bayes' rule:**

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}$$

Bayes' Rule

Similarly, we have:

$$p(x, y, z) = p(x|y, z)p(y, z) = p(y|x, z)p(x, z) = p(y, x, z)$$

Solving for $p(x|y, z)$ yields:

$$p(x|y, z) = \frac{p(y|x, z)p(x, z)}{p(y, z)} = \frac{p(y|x, z)p(x|z)p(z)}{p(y|z)p(z)} = \frac{p(y|x, z)p(x|z)}{p(y|z)}$$

In other words, Bayes' rule also holds when conditioning on another random variable z.

The Bayes Filter

The Bayes Filter

Recursive Bayesian Estimation

- We assume that we have only one object which we want to track
- Further, assume that exactly one noisy observation is available per frame
- We want to do probabilistic state inference, i.e., estimate distributions
- Recursive Bayesian estimation (=Bayes filter) is a probabilistic approach for estimating an unknown probability density function recursively over time using
  - Incoming measurements (e.g., detected object location)
  - A system process model (e.g., constant velocity motion model)
- Two alternating steps:
  1. Prediction: Predict where the object should be in the next frame
  2. Correction: Correct prediction based on current observation
- Remark: A good motion model is also useful for association $\Rightarrow$ Unit 3

The Bayes Filter

General assumptions:

- Camera is not moving instantly to a new viewpoint
- Objects do not disappear and reappear in different places
- Gradual change in pose between camera and scene

Assumptions on motion and observation model:

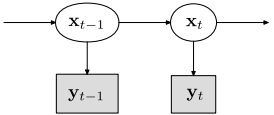
- The Bayes filter assumes that the state follows a Markov process
- A Kalman filter assumes linear and Gaussian motion/observation models
- More advanced models (EKF, UKF, Particle Filter) are less restrictive

The Bayes Filter

Formal Definition:

- Let $\mathbf{x}_t \in \mathbb{R}^M$ denote the true **hidden** state of the system at time t
- Let $\mathbf{y}_t \in \mathbb{R}^N$ denote some noisy measurement of $\mathbf{x}_t$ at time t
- The Bayes filter assumes that $\mathbf{X} = \{\mathbf{x}_t\}$ is an unobserved **Markov process** and $\mathbf{Y} = \{\mathbf{y}_t\}$ are the corresponding measurements:

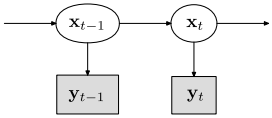
$$p(\mathbf{X}, \mathbf{Y}) = p(\mathbf{x}_1) \prod_{t=2}^{T} p(\mathbf{x}_t|\mathbf{x}_{t-1}) \prod_{t=1}^{T} p(\mathbf{y}_t|\mathbf{x}_t)$$



- Here, $p(\mathbf{x}_1)$ is the prior distribution over the state at the beginning of the sequence
- Goal: Perform inference (estimate $\mathbf{X}$ from $\mathbf{Y}$) in this **Hidden Markov Model**

The Bayes Filter

$$p(\mathbf{X}, \mathbf{Y}) = p(\mathbf{x}_1) \prod_{t=2}^{T} p(\mathbf{x}_t|\mathbf{x}_{t-1}) \prod_{t=1}^{T} p(\mathbf{y}_t|\mathbf{x}_t)$$



- For **recursive state estimation,** we take advantage of **Bayes rule:**

$$\underbrace{p(\mathbf{x}_t|\mathbf{y}_{1:t})}_{\text{Posterior}} = \frac{p(\mathbf{y}_t|\mathbf{x}_t, \mathbf{y}_{1:t-1})\, p(\mathbf{x}_t|\mathbf{y}_{1:t-1})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})} \propto \underbrace{p(\mathbf{y}_t|\mathbf{x}_t)}_{\text{Observation}} \underbrace{p(\mathbf{x}_t|\mathbf{y}_{1:t-1})}_{\text{Prediction}}$$

- The **predictive distribution** can be further decomposed as:

$$\begin{aligned}
\underbrace{p(\mathbf{x}_t|\mathbf{y}_{1:t-1})}_{\text{Prediction}} &= \int p(\mathbf{x}_t, \mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})\, d\mathbf{x}_{t-1} \\
&= \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_{1:t-1})\, p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})\, d\mathbf{x}_{t-1} \\
&= \int \underbrace{p(\mathbf{x}_t|\mathbf{x}_{t-1})}_{\text{Motion Model}} \underbrace{p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})}_{\text{Prev. Posterior}}\, d\mathbf{x}_{t-1}
\end{aligned}$$

The Bayes Filter

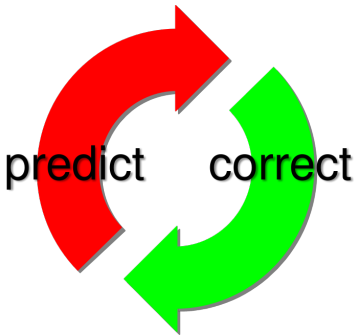
We obtain a **recursive state estimation algorithm:**

1. **Prediction**

$$\underbrace{p(\mathbf{x}_t|\mathbf{y}_{1:t-1})}_{\text{Prediction}} = \int \underbrace{p(\mathbf{x}_t|\mathbf{x}_{t-1})}_{\text{Motion Model}} \underbrace{p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})\, d\mathbf{x}_{t-1}}_{\text{Prev. Posterior}}$$

2. **Correction**

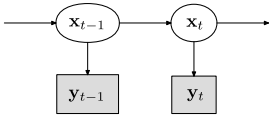
$$\underbrace{p(\mathbf{x}_t|\mathbf{y}_{1:t})}_{\text{Posterior}} \propto \underbrace{p(\mathbf{y}_t|\mathbf{x}_t)}_{\text{Observation}} \underbrace{p(\mathbf{x}_t|\mathbf{y}_{1:t-1})}_{\text{Prediction}}$$



The Bayes filter runs **online** (i.e., update per frame).

The Kalman Filter

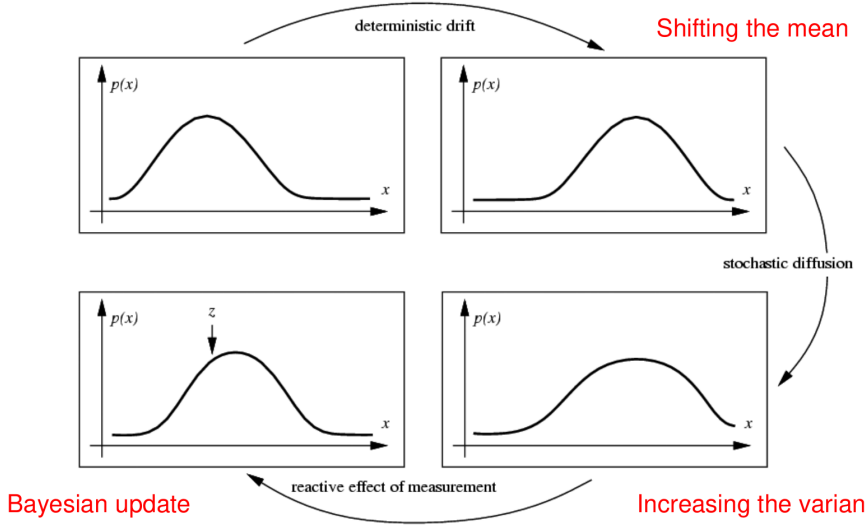
$$p(\mathbf{X}, \mathbf{Y}) = p(\mathbf{x}_1) \prod_{t=2}^{T} p(\mathbf{x}_t|\mathbf{x}_{t-1}) \prod_{t=1}^{T} p(\mathbf{y}_t|\mathbf{x}_t)$$



- If $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ and $p(\mathbf{y}_t|\mathbf{x}_t)$ are linear and normally distributed, we obtain the **Kalman filter** (KF) $\Rightarrow$ Parameters: mean $\boldsymbol{\mu}$ / covariance $\boldsymbol{\Sigma}$
- The solution for the Kalman filter is given in closed form as products/marginals are **Gaussian** $\Rightarrow$ max. likelihood = least squares (omitting formulas here)
- The Kalman filter is the optimal linear filter in least squares sense
- Non-linear case: **Extended Kalman Filter** (EKF), **Unscented Kalman Filter** (UKF)
- Multiple modes can be handled using a **Particle Filter** which computes a sampling-based approximation to the state posterior $p(\mathbf{x}_t|\mathbf{y}_{1:t})$
- More recently, learning-based motion models (e.g., RNN) have been explored

https://en.wikipedia.org/wiki/Kalman_filter

Density Propagation



Examples: Linear Cases (1D)

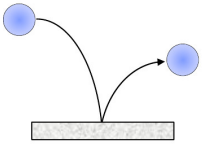
Constant Velocity: (state vector: position/velocity $\mathbf{x}_t = (p_t, v_t)^\top$)

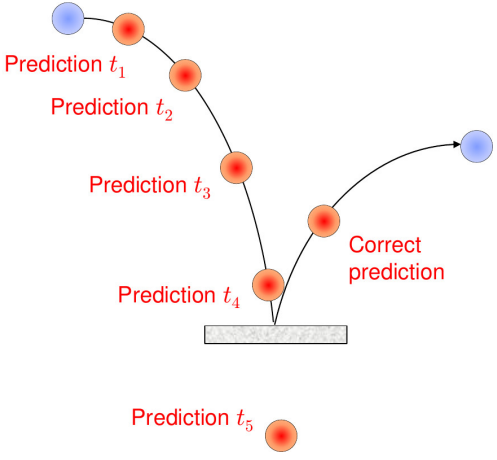
$$\mathbf{x}_t = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \mathbf{x}_{t-1} + \underbrace{\boldsymbol{\epsilon}_m}_{\sim\mathcal{N}(\mathbf{0},\boldsymbol{\Sigma}_m)} \qquad y_t = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x}_t + \underbrace{\boldsymbol{\epsilon}_o}_{\sim\mathcal{N}(\mathbf{0},\boldsymbol{\Sigma}_o)}$$

Constant Acceleration: (state vector: position/velocity/acceleration $\mathbf{x}_t = (p_t, v_t, a_t)^\top$)

$$\mathbf{x}_t = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}(\Delta t)^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x}_{t-1} + \underbrace{\boldsymbol{\epsilon}_m}_{\sim\mathcal{N}(\mathbf{0},\boldsymbol{\Sigma}_m)} \qquad y_t = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \mathbf{x}_t + \underbrace{\boldsymbol{\epsilon}_o}_{\sim\mathcal{N}(\mathbf{0},\boldsymbol{\Sigma}_o)}$$

Examples: Non-linear Cases



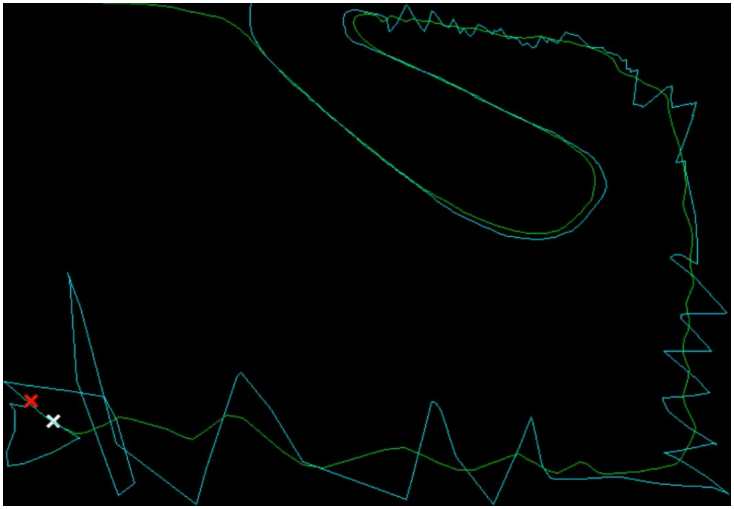


- Many interesting cases don't have linear dynamics (pedestrian, bouncing ball)
- A constant acceleration model would predict too far from true position

Applications: Single Object Tracking

https://www.youtube.com/watch?v=9CBDH1Y4Iw0

Applications: Single Object Tracking



https://www.youtube.com/watch?v=NYdwpX1a7-Y

Applications: Moon Landing

”The Kalman filter was used, for example, in the Ranger, Mariner, and Apollo missions of the 1960s. In particular, the on-board computer that guided the descent of the Apollo 11 lunar module to the moon had a Kalman filter. According to William Lear, an aerospace engineer, NASA contacted him about nine months before Apollo 11’s scheduled launch because their Earth-based tracking program wasn’t working. Lear wrote a 21-state Kalman filter program, which went into the Doppler radar system. The final check of the program, Lear recalls, was done **the day before** Armstrong, Aldrin, and Collins took off.”

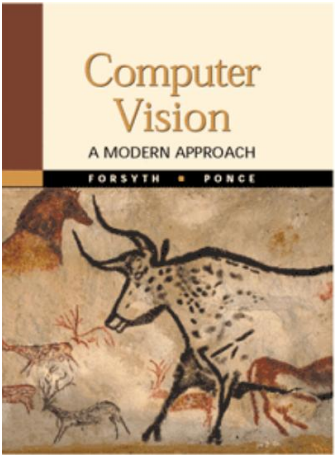
Engineers Look to Kalman Filtering for Guidance, Barry Cipra, SIAM News 1993

https://www.cs.unc.edu/~welch/kalman/siam_cipra.html

The Kalman Filter

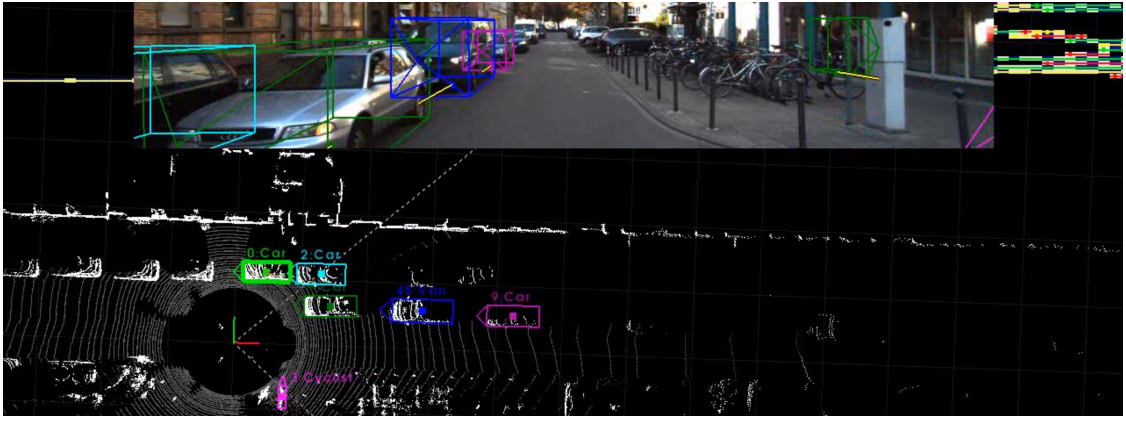
Further Readings:

- Forsyth and Ponce:
  Computer Vision – A Modern Approach
  Prentice Hall, 2003
- Good introduction to tracking
  and Kalman filters in Chapter 17
- Wikipedia:
  `https://en.wikipedia.org/wiki/Kalman_filter`



11.3
Association

Multi-Object Tracking



- In self-driving, we typically have to track **multiple objects** at the same time
- How can we **associate detections** in a new frame to existing object tracks?

Multi-Object Tracking

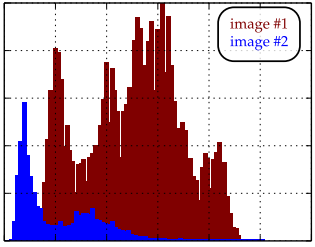
Algorithm

1. Predict objects from previous frame and detect objects in current frame
2. Associate detections to object tracks (initiate/delete tracks if necessary)
3. Correct predictions with observations (e.g., Kalman Filter)

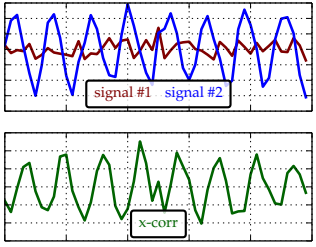
When do observations in consecutive frames belong together?

- Predict bounding box (via motion model) and measure overlap
- Compare color histograms or normalized cross-correlation
- Estimate optical flow and measure agreement
- Compare relative location and size of bounding box
- Compare orientation of detected objects

Object Association Measures

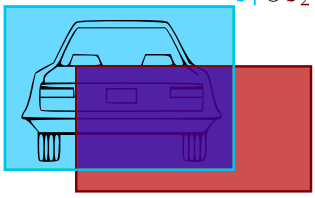


(a) Color Histogram Similarity

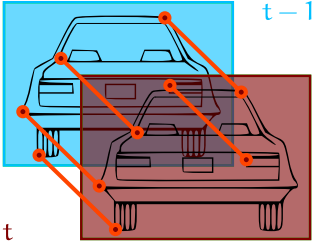


(b) Cross Correlation

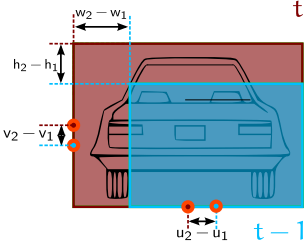
$$s_O = \frac{\mathbf{b}_1 \cap \mathbf{b}_2}{\mathbf{b}_1 \cup \mathbf{b}_2}$$



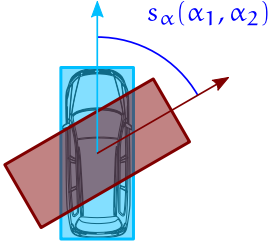
(c) Bounding Box Overlap



(d) Optical Flow Overlap



(e) Positional and Size Similarity



(f) Orientation Similarity

Metric Learning

Anchor (A)

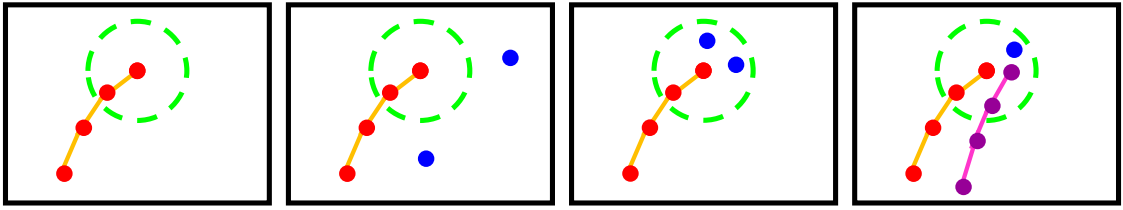
Positive (P)

Negative (N)

- Idea: use convolutional networks to **learn an object representation** $f_\theta(I)$
- A good representation maps images of the same object to **similar features**
- We want: $\|f_\theta(A) - f_\theta(P)\|^2 < \|f_\theta(A) - f_\theta(N)\|^2$ ($\Rightarrow$ learn a ranking)
- **Triplet loss:** $\mathcal{L} = \max(0, \|f_\theta(A) - f_\theta(P)\|^2 - \|f_\theta(A) - f_\theta(N)\|^2 + m)$
- The margin m ensures that the loss focuses on the difficult cases
- Improve training: Hard-negative mining, intelligent sampling, group loss

Elezi, Vascon, Torcinovich, Pelillo and Leal-Taixé: The Group Loss for Deep Metric Learning. ECCV, 2020.

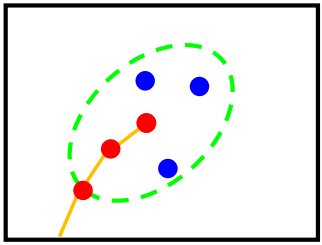
Correspondence Ambiguities



- Predictions may not be supported by measurements
  Have objects ceased to exist, are they occluded or did detection fail?
- There may be unexpected measurements
  Newly visible objects or just detector noise?
- More than one measurement may match a prediction
  Which measurement is the correct one (what about the others)?
- A measurement may match multiple predictions
  Which object shall the measurement be assigned to?

Nearest Neighbor Association

- Only consider measurements within a certain **gating area** around the prediction
- Associate detection $\mathbf{y}_t \in \mathcal{Y}_t$ **closest to prediction:**



$$\underset{\mathbf{y}_t \in \mathcal{Y}_t}{\operatorname{argmin}} \ \|\mathbf{y}_t - \mathbf{x}_t\|_2 \quad = \quad \underset{\mathbf{y}_t \in \mathcal{Y}_t}{\operatorname{argmin}} \ \sqrt{(\mathbf{y}_t - \mathbf{x}_t)^\top (\mathbf{y}_t - \mathbf{x})}$$

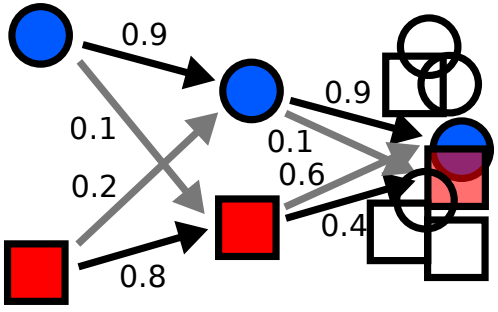
- Better: If a Gaussian prediction model $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ is used (e.g., Kalman Filter), consider the detection **most likely under the prediction model:**

$$\underset{\mathbf{y}_t \in \mathcal{Y}_t}{\operatorname{argmax}} \ \mathcal{N}(\mathbf{y}_t|\mathbf{x}_t, \mathbf{\Sigma}_t) \quad = \quad \underset{\mathbf{y}_t \in \mathcal{Y}_t}{\operatorname{argmin}} \ \underbrace{\sqrt{(\mathbf{y}_t - \mathbf{x}_t)^\top \mathbf{\Sigma}_t^{-1} (\mathbf{y}_t - \mathbf{x})}}_{\text{Mahalanobis Distance}}$$

- The Mahalanobis distance can also be used to define the gating area

Nearest Neighbor Association

What is the problem with this approach? Let's consider a simple example:

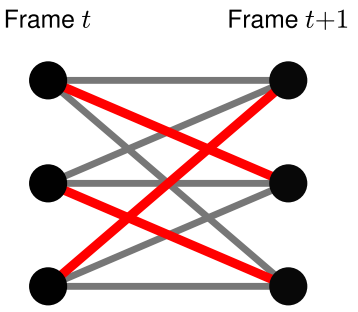


- Numbers represent **association scores:** higher is better/more similar
- **Choosing the nearest neighbor often fails** due to ambiguities
- In practice, many noisy detections! When to continue, start or kill a track?

Bipartite Graph Matching

Bipartite Graph Matching

- Given a bipartite graph, a matching is a subset of the edges for which every vertex belongs to exactly one of the edges
- Frame-to-frame tracking can be formulated as **bipartite graph matching** problem (chooses at most one match in each row and column while maximizing score)



Frame t \ Frame $t+1$			
	0.11	0.95	0.23
	0.85	0.25	0.89
	0.90	0.12	0.81

Bipartite Graph Matching

Formal Definition:

- Assume N detections in the previous and M detections in the current frame
- We can construct a $M \times N$ **table of matching scores**
- **Task:** Choose 1:1 matching that **maximizes the sum of scores**

	1	**2**	**3**	**4**	**5**
1	0.95	0.76	0.62	0.41	0.06
2	0.23	0.46	0.79	0.94	0.35
3	0.61	0.02	0.92	0.92	0.81
4	0.49	0.82	0.74	0.41	0.01
5	0.89	0.44	0.18	0.89	0.14

- What is the optimal 1:1 matching?

Bipartite Graph Matching

Formal Definition: Given an $M \times N$ matrix of association scores $\mathbf{S}$, determine a binary $M \times N$ matrix $\mathbf{Z}$ that maximizes the total score:

$$\underset{\mathbf{Z}}{\operatorname{argmax}} \sum_{i=1}^{M} \sum_{j=1}^{N} s_{ij}\, z_{ij}$$

$$s.t. \quad \forall_j : \sum_{i=1}^{M} z_{ij} = 1 \quad \forall_i : \sum_{j=1}^{N} z_{ij} = 1 \quad z_{ij} \in \{0, 1\}$$

- This is an integer linear program (ILP), but it is not NP hard as we will see
- The permutation matrix constraints ensure that we can only choose one number from each row and from each column (1:1 matching)

Bipartite Graph Matching

	1	2	3	4	5
1	0.95	0.76	0.62	0.41	0.06
2	0.23	0.46	0.79	0.94	0.35
3	0.61	0.02	0.92	0.92	0.81
4	0.49	0.82	0.74	0.41	0.01
5	0.89	0.44	0.18	0.89	0.14

- How many different assignments are possible?
  - $5 \times 4 \times 3 \times 2 \times 1 = 120 \quad (N!)$
- In practice, we often have to deal with 10-100 detections
- Exhaustive search becomes intractable very quickly

Greedy Matching

Greedy Algorithm:

- Start with an unmarked matrix
- For i = 1 to N:
  - Mark largest value that isn't in a row/column with already marked entries

	1	2	3	4	5
1	0.95	0.76	0.62	0.41	0.66
2	0.23	0.46	0.79	0.94	0.35
3	0.61	0.02	0.92	0.92	0.81
4	0.49	0.82	0.74	0.41	0.01
5	0.89	0.44	0.18	0.89	0.14

Greedy Matching

Greedy Algorithm:

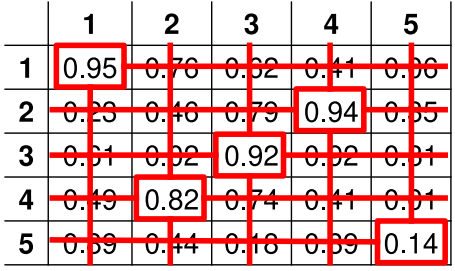
- Start with an unmarked matrix
- For i = 1 to N:
  - Mark largest value that isn't in a row/column with already marked entries

	1	2	3	4	5
1	0.95	0.76	0.62	0.41	0.66
2	0.23	0.46	0.79	0.94	0.35
3	0.61	0.02	0.92	0.92	0.81
4	0.49	0.82	0.74	0.41	0.01
5	0.89	0.44	0.18	0.89	0.14

Greedy Matching

Greedy Algorithm:

- Start with an unmarked matrix
- For i = 1 to N:
  - Mark largest value that isn't in a row/column with already marked entries



Greedy vs. Optimal Solution

	1	2	3	4	5
1	0.95	0.76	0.62	0.41	0.06
2	0.23	0.46	0.79	0.94	0.35
3	0.61	0.02	0.92	0.92	0.81
4	0.49	0.82	0.74	0.41	0.01
5	0.89	0.44	0.18	0.89	0.14

Greedy Solution: Score = 3.77

	1	2	3	4	5
1	0.95	0.76	0.62	0.41	0.06
2	0.23	0.46	0.79	0.94	0.35
3	0.61	0.02	0.92	0.92	0.81
4	0.49	0.82	0.74	0.41	0.01
5	0.89	0.44	0.18	0.89	0.14

Optimal Solution: Score = 4.26

- Greedy matching is easy to implement, quick to run and finds good solutions
- However, greedy matching often does not yield the optimal solution
- **Optimal solution:** Hungarian method $O(N^4)$ or Ford-Fulkerson $O(N^3)$

Kuhn: The Hungarian Method for the assignment problem. Naval Research Logistics Quarterly, 2: 83–97, 1955.

Handling Missing Tracks and Detections

	1	2	3	4
1	0.95	0.76	0.62	0.41
2	0.23	0.46	0.79	0.94
3	0.01	0.02	0.06	0.01
4	0.49	0.82	0.74	0.41

Score Matrix

	1	2	3	4	
1	0.95	0.76	0.62	0.41	0.30
2	0.23	0.46	0.79	0.94	0.30
3	0.01	0.02	0.06	0.01	0.30
4	0.49	0.82	0.74	0.41	0.30
	0.30	0.30	0.30	0.30	0.30

Augmented Matrix

- Typically there will be a **different number of tracks and observations**
- Moreover, object tracks may end (no detection) or start (no existing track)
- Solution: Introduce **extra nodes** to absorb matches (example: track 3 ends)

Kuhn: The Hungarian Method for the assignment problem. Naval Research Logistics Quarterly, 2: 83–97, 1955.

Graph-based Tracking

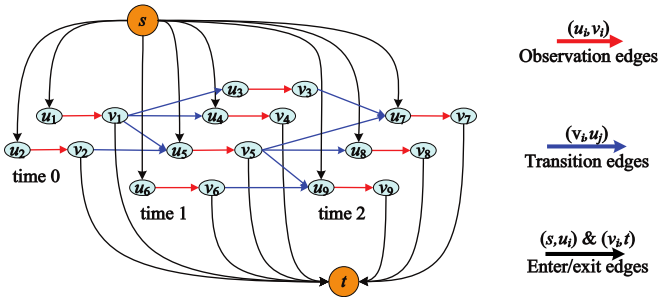
Multi-Frame Tracking as Graph Optimization

- We have seen how we can solve the association problem for 2 frames
- This is not necessarily optimal for longer sequences
- In particular, wrong associations cannot be “undone” at later frames
- Let’s now look at the multi-frame case
- Setting: Tracking-by-detection (as before)
- Goal: Solve for all associations across all frames of a sequence
- Approach: Cast as min-cost flow network problem [Zhang et al., CVPR 2008]

Zhang, Li and Nevatia: Global data association for multi-object tracking using network flows. CVPR, 2008.

Multi-Frame Tracking as Graph Optimization

Can be formulated as Min-Cost Flow Problem:



Goal: Push flow from source to sink such that it generates minimal cost

- Observation edges carry negative cost (proportional to detection confidence)
- Transition edges carry positive cost (to separate dissimilar objects)

Zhang, Li and Nevatia: Global data association for multi-object tracking using network flows. CVPR, 2008.

Multi-Frame Tracking as Graph Optimization

Mathematical formulation:

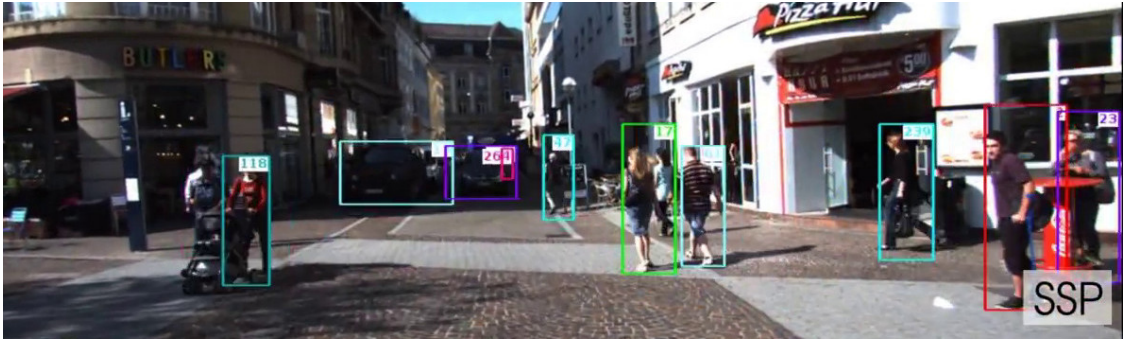
$$\underset{\mathbf{f}}{\operatorname{argmin}} \sum_i c_i^{en} f_i^{en} + \sum_i c_i^{ex} f_i^{ex} + \sum_{i,j} c_{i,j}^{li} f_{i,j}^{li} + \sum_i c_i^{det} f_i^{det}$$

$$s.t.\ f_i^{en} + \sum_j f_{j,i}^{li} = f_i^{det} = f_i^{ex} + \sum_j f_{i,j}^{li} \quad \forall i$$

- $f_i \in \{0, 1\}$: Binary flow variables
- $c_i \in \mathbb{R}$: Cost for entry/exit/link/detection
- Constraints on flow conservation $\Rightarrow$ no overlapping object trajectories

Zhang, Li and Nevatia: Global data association for multi-object tracking using network flows. CVPR, 2008.

Results on KITTI

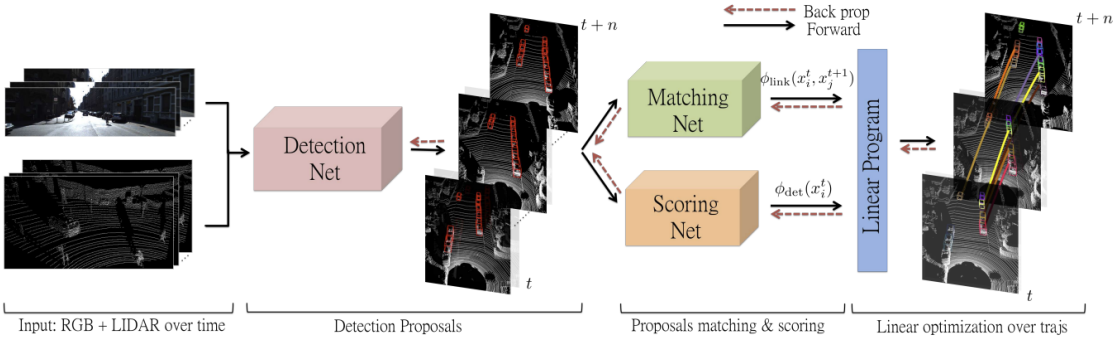


- Optimal solution in polynomial time via **successive shortest path algorithm**
- But **only pairwise relationships** ⇒ no occlusions, simplistic motion model
- Batch processing ⇒ **online versions** have been proposed [Lenz et al., ICCV 2015]

Lenz, Geiger and Urtasun: FollowMe: Efficient Online Min-Cost Flow Tracking with Bounded Memory and Computation. ICCV, 2015.

Advanced Graph-based Models

Differentiable Graph Optimization

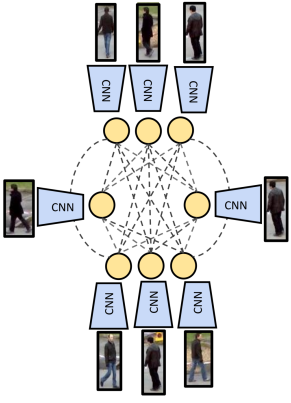


- Train detection and tracking jointly by **backpropagating** through a linear program

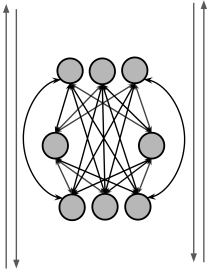
Frossard and Urtasun: End-to-end Learning of Multi-sensor 3D Tracking by Detection. ICRA, 2018.

Learning a Neural Solver

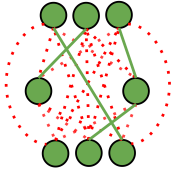
Input



Graph Construction + Feature Encoding



Neural Message Passing



Edge Classification

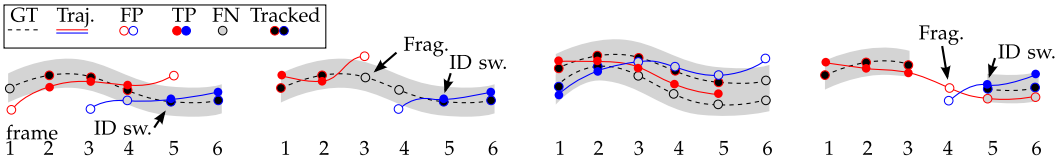
Output

- End-to-end formulation using a **graph neural network** (but batch processing)

Brasó and Leal-Taixé: Learning a Neural Solver for Multiple Object Tracking. CVPR, 2020.

Multi-Object Tracking Evaluation

Multi-Object Tracking Evaluation



- Associate prediction with ground truth tracks (bipartite graph matching)
- **MOTA:** Multiple Object Tracking Accuracy

$$MOTA = 1 - \frac{\sum_t FN_t + FP_t + IDSW_t}{\sum_t GT_t}$$

- FN/FP: False negative/positive detections, IDSW: ID switches, GT: GT objects
- **HOTA:** A Higher Order Metric for Evaluating Multi-object Tracking
  `https://github.com/JonathonLuiten/TrackEval`

KITTI Benchmark

http://www.cvlibs.net/datasets/kitti/eval_tracking.php

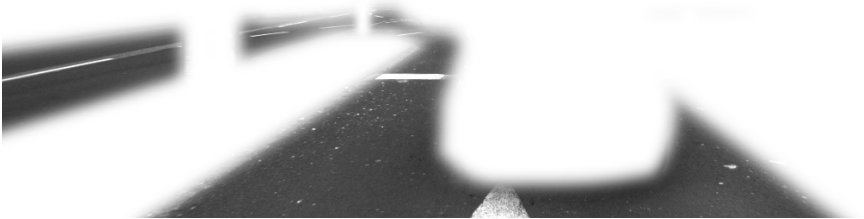
Geiger, Lenz and Urtasun: Are we ready for autonomous driving? The KITTI vision benchmark suite. CVPR, 2012.

11.4
Holistic Scene Understanding

Lane Detection vs. Intersection Understanding

Aly: Real time Detection of Lane Markers in Urban Streets. IV, 2008.

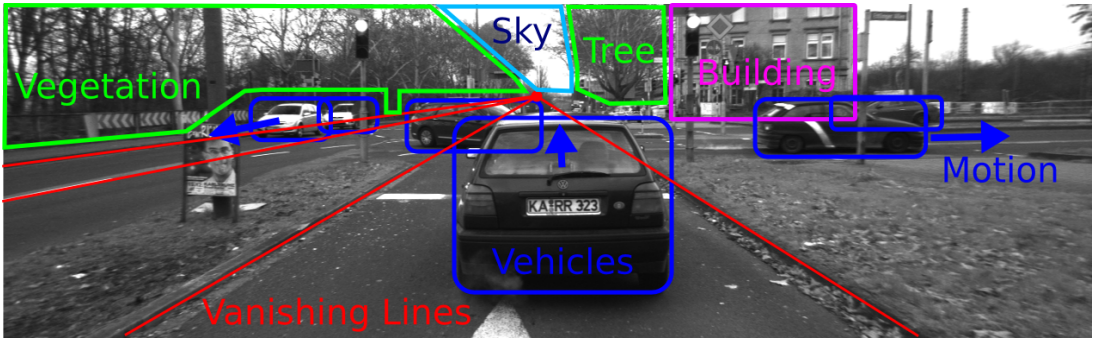
Lane Detection vs. Intersection Understanding



Intersection from a Lane Detector's Perspective

- We must fuse all cues to obtain a complete/holistic understanding
- Fusing multiple cues leads to more robust estimates

Lane Detection vs. Intersection Understanding



Intersection from a Human Perspective

- We must fuse all cues to obtain a complete/holistic understanding
- Fusing multiple cues leads to more robust estimates

Sensor Fusion – A Simple Example

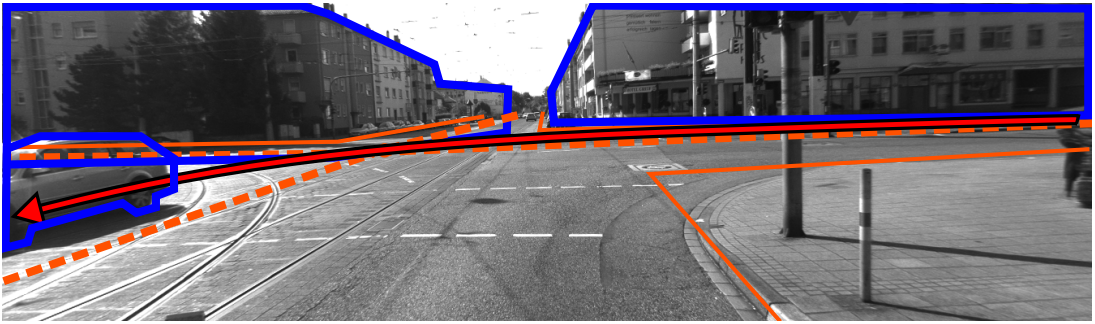
- Suppose we are given an object detection by 2 sensors (camera and lidar)
- Let $\mathbf{x} \in \mathbb{R}^2$ denote the (unknown) object location
- Let $\mathbf{z}_1 \in \mathbb{R}^2$ and $\mathbf{z}_2 \in \mathbb{R}^2$ be the noisy measurements of the camera and lidar
- The **posterior probability distribution** over the object location is given by

$$p(\mathbf{x}|\mathbf{z}_1, \mathbf{z}_2) = \frac{p(\mathbf{z}_1, \mathbf{z}_2|\mathbf{x})\, p(\mathbf{x})}{p(\mathbf{z}_1, \mathbf{z}_2)} \propto p(\mathbf{z}_1|\mathbf{x})\, p(\mathbf{z}_2|\mathbf{x})\, p(\mathbf{x})$$

- Assuming a uniform (uninformative) prior we obtain

$$p(\mathbf{x}|\mathbf{z}_1, \mathbf{z}_2) \propto p(\mathbf{z}_1|\mathbf{x})\, p(\mathbf{z}_2|\mathbf{x})$$

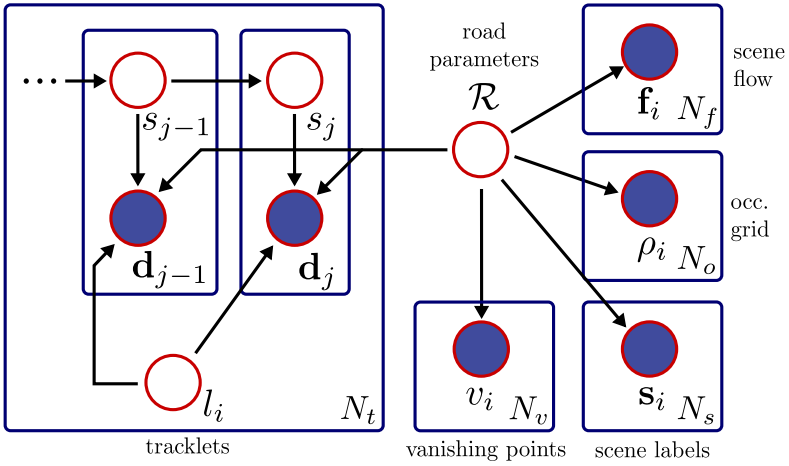
3D Traffic Scene Understanding



- **Goal:** Infer from short stereo video sequences (no maps/lidar here)
  - **Topology** and **geometry** of the scene
  - **Semantic information** (traffic situation)
- Probabilistic generative model of urban scenes, fusing many different cues ...

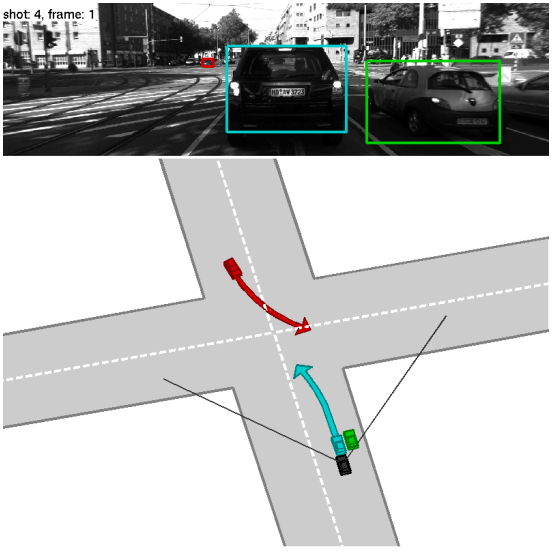
Geiger, Lauer, Wojek, Stiller and Urtasun: 3D Traffic Scene Understanding From Movable Platforms. PAMI, 2014.

Probabilistic Graphical Model



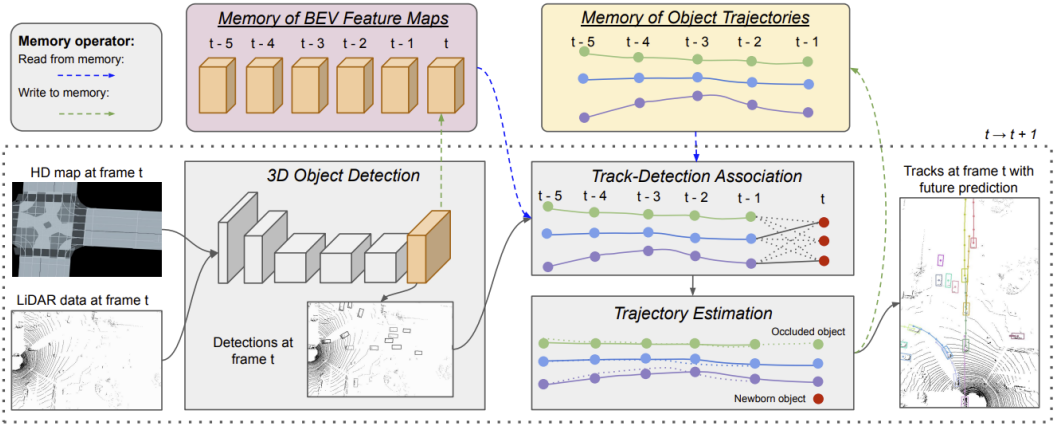
Geiger, Lauer, Wojek, Stiller and Urtasun: 3D Traffic Scene Understanding From Movable Platforms. PAMI, 2014.

Experimental Results



Geiger, Lauer, Wojek, Stiller and Urtasun: 3D Traffic Scene Understanding From Movable Platforms. PAMI, 2014.

PnPNet: Perception and Prediction with Tracking in the Loop



- Joint perception, tracking and motion forecasting / trajectory estimation

Liang, Yang, Zeng, Chen, Hu, Casas and Urtasun: PnPNet: End-to-End Perception and Prediction With Tracking in the Loop. CVPR, 2020.

Argoverse: 3D Tracking and Forecasting With Rich Maps

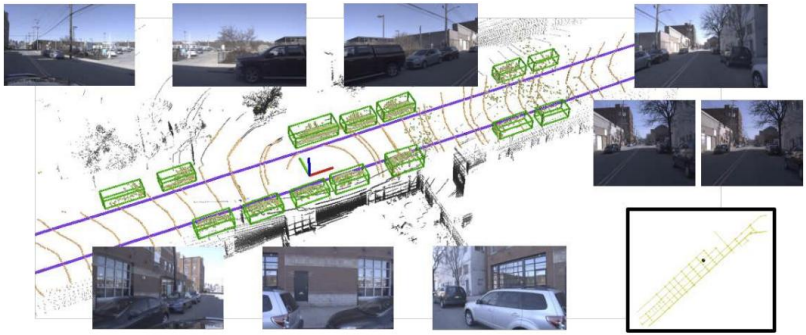


Figure 1: We introduce a dataset for 3D tracking and forecasting with *rich maps* for autonomous driving. Our dataset contains sequences of LiDAR measurements, 360° RGB video, front-facing stereo (middle-right), and 6-dof localization. All sequences are aligned with maps containing lane center lines (magenta), driveable region (orange), and ground height. Sequences are annotated with 3D cuboid tracks (green). A wider map view is shown in the bottom-right.

`https://www.argoverse.org/`

Summary

- Object tracking requires detection, association and filtering
- Precise knowledge of the observation/motion model improves tracking
- Offline tracking requires future observations $\Rightarrow$ focus on online tracking
- Filtering is the task of estimating the state given a sequence of observations
- The Bayes filter comprises 2 steps: prediction and correction
- The Kalman filter is a special case (Gauss linear) of the Bayes filter
- Frame-to-frame association can be solved using bipartite graph matching
- Multi-frame tracking can be formulated as a min-cost flow problem
- Multiple Object Tracking Accuracy (MOTA) is often used to evaluate performance
- Joint estimation of object tracks and scene layout can improve performance