Object tracking

Boris Lestsov

Goal

1) Build association between bboxes from frame to frame on video.

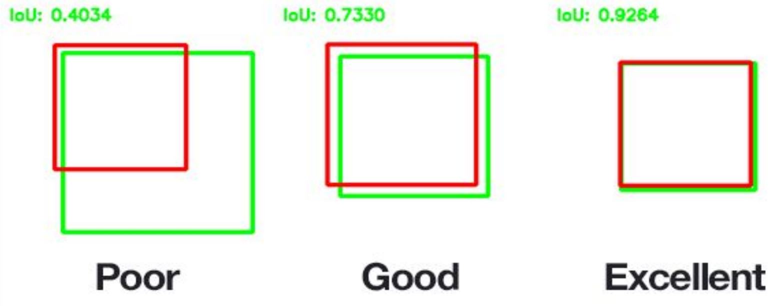
We give each detected bbox an id.

Goal

1) Build association between bboxes from frame to frame on video.
2) Fix detector errors: false positive detections and short false negatives.

We give each detected bbox an id.

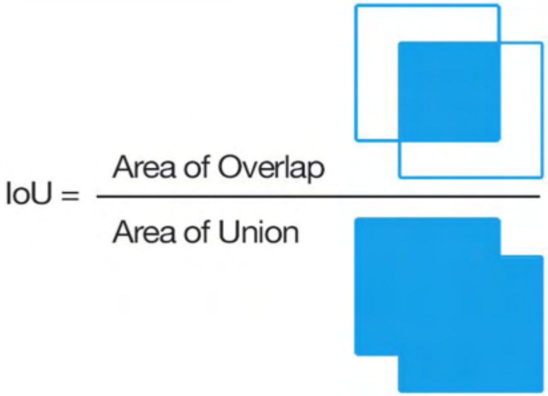
Association by IoU



IoU formula and images.

Given N tracks and M boxes, compute IoU matrix NxM => CostMatrix is 1 - IoU.

Use hungarian algorithm to find best association between boxes and tracks. Initiate new tracks and delete old ones.



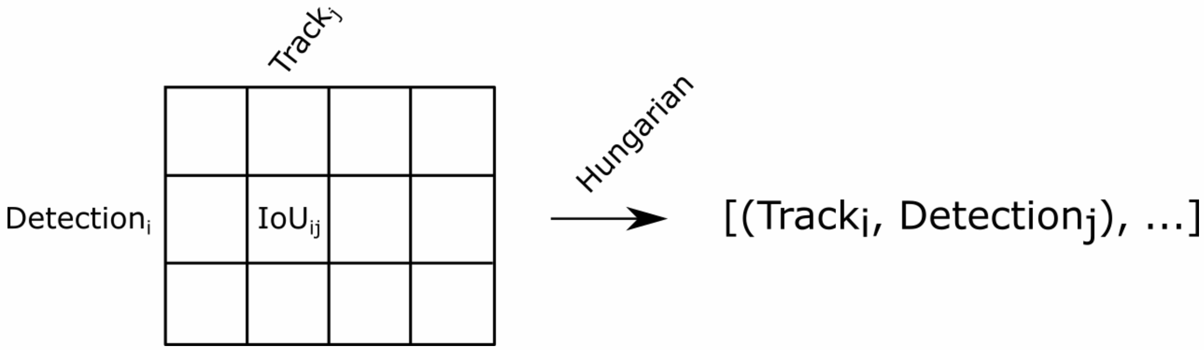
Hungarian algorithm:

N workers, N jobs => best assignment between workers and jobs in polynomial time.

Association by IoU

Hungarian algorithm:

N workers, N jobs => best assignment between workers and jobs in polynomial time.

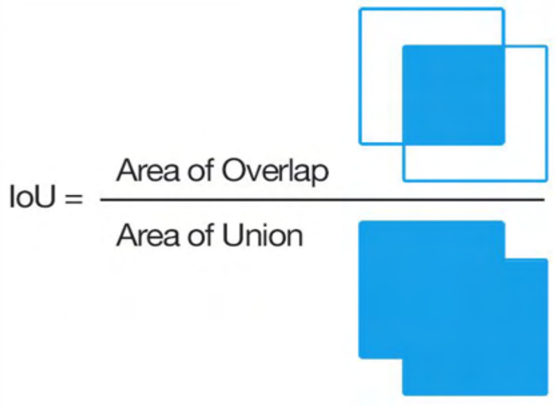


Association by IoU

IoU formula and images.

Given N tracks and M boxes, compute IoU matrix NxM => CostMatrix is 1 - IoU.

Use hungarian algorithm to find best association between boxes and tracks. Initiate new tracks and delete old ones.



Hungarian algorithm:

N workers, N jobs => best assignment between workers and jobs in polynomial time.

Problem: objects move on video => their boxes move => let's try to predict their movement on the next frame.

Kalman Filter

Very general and widely used in diverse applications: self-driving cars, drones, robotics, spaceships, aviation, economics, thermodynamics, …

Data fusion: continuously fuse noisy model prediction and noisy measurements.

Kalman Filter: Outline

Object is expressed as a vector of parameters (**state vector**).

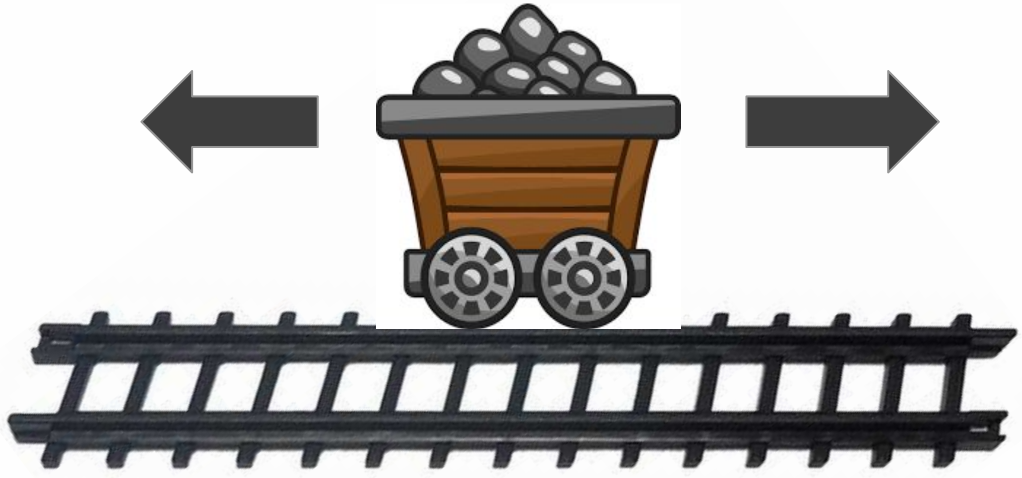
Iterate over time:

1) Predict state vector at the next timestep.
2) Correct the prediction using measurements.

Kalman filter gives MLE of parameter values (minimizes MSE in case of gaussians).

Kalman Filter: Example

One dimensional cart that can move along one axis.



Kalman Filter: Model

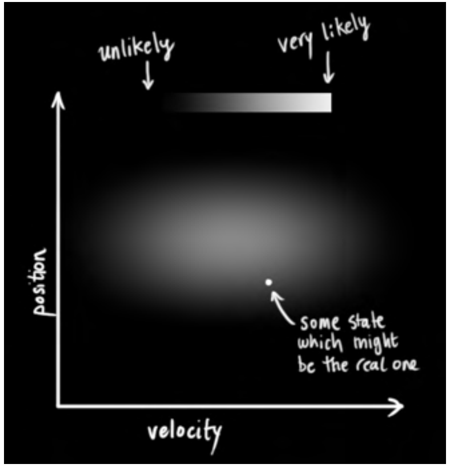
$$\hat{\mathbf{x}}_k = \begin{bmatrix} \text{position} \\ \text{velocity} \end{bmatrix}$$

$$\mathbf{P}_k = \begin{bmatrix} \Sigma_{pp} & \Sigma_{pv} \\ \Sigma_{vp} & \Sigma_{vv} \end{bmatrix}$$

Express information about object as a normal distribution.

P_k - **covariance matrix**, our uncertainty about state vector.

In example: position and velocity can be correlated => matrix is not diagonal.



Source: https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/

Kalman Filter: Model

Express state transformation with **prediction matrix**.
Depends on environment.

The modelled process is linear.

Our example: cart moves with some speed => its position changes.

Source: https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/

$$\begin{aligned} p_k &= p_{k-1} + \Delta t v_{k-1} \\ v_k &= \phantom{p_{k-1} + \Delta t} v_{k-1} \end{aligned}$$

$$\begin{aligned} \hat{\mathbf{x}}_k &= \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \hat{\mathbf{x}}_{k-1} \\ &= \mathbf{F}_k \hat{\mathbf{x}}_{k-1} \end{aligned}$$

$$\begin{aligned} Cov(x) &= \Sigma \\ Cov(\mathbf{A}x) &= \mathbf{A}\Sigma\mathbf{A}^T \end{aligned}$$

$$\begin{aligned} \hat{\mathbf{x}}_k &= \mathbf{F}_k \hat{\mathbf{x}}_{k-1} \\ \mathbf{P}_k &= \mathbf{F_k} \mathbf{P}_{k-1} \mathbf{F}_k^T \end{aligned}$$

Kalman Filter: External Control

u_k - **control vector**.

B_k - **control matrix**.

Our example: an external force is applied to the cart.

$$
\begin{aligned}
p_k &= p_{k-1} + \Delta t v_{k-1} + \frac{1}{2} a \Delta t^2 \\
v_k &= \qquad\qquad\; v_{k-1} + a \Delta t
\end{aligned}
$$

$$
\begin{aligned}
\hat{\mathbf{x}}_k &= \mathbf{F}_k \hat{\mathbf{x}}_{k-1} + \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} a \\
&= \mathbf{F}_k \hat{\mathbf{x}}_{k-1} + \mathbf{B}_k \vec{\mathbf{u}_k}
\end{aligned}
$$

Source: https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/

Kalman Filter: External Noise

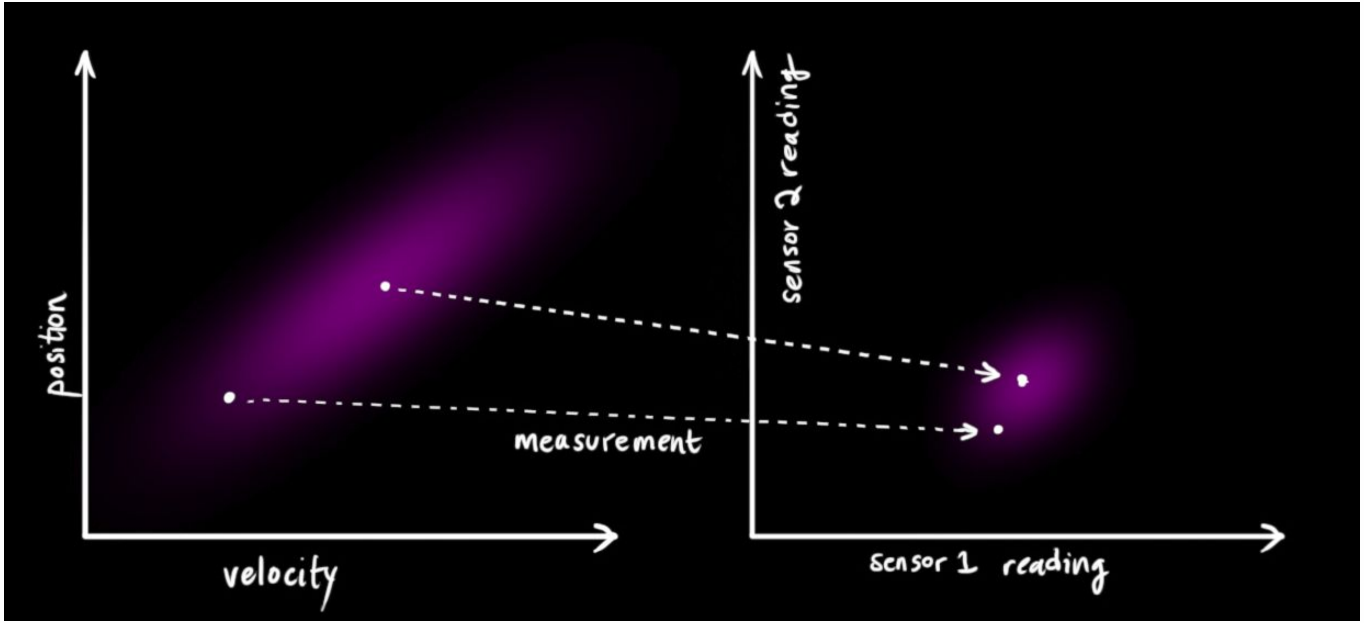
The environment might itself might cause uncertainty => add matrix Q_k - **model noise covariance matrix**.

$$\hat{\mathbf{x}}_k = \mathbf{F}_k \hat{\mathbf{x}}_{k-1} + \mathbf{B}_k \vec{\mathbf{u}_k}$$
$$\mathbf{P}_k = \mathbf{F}_k \mathbf{P}_{k-1} \mathbf{F}_k^T + \mathbf{Q}_k$$

Our example: friction adds noise.

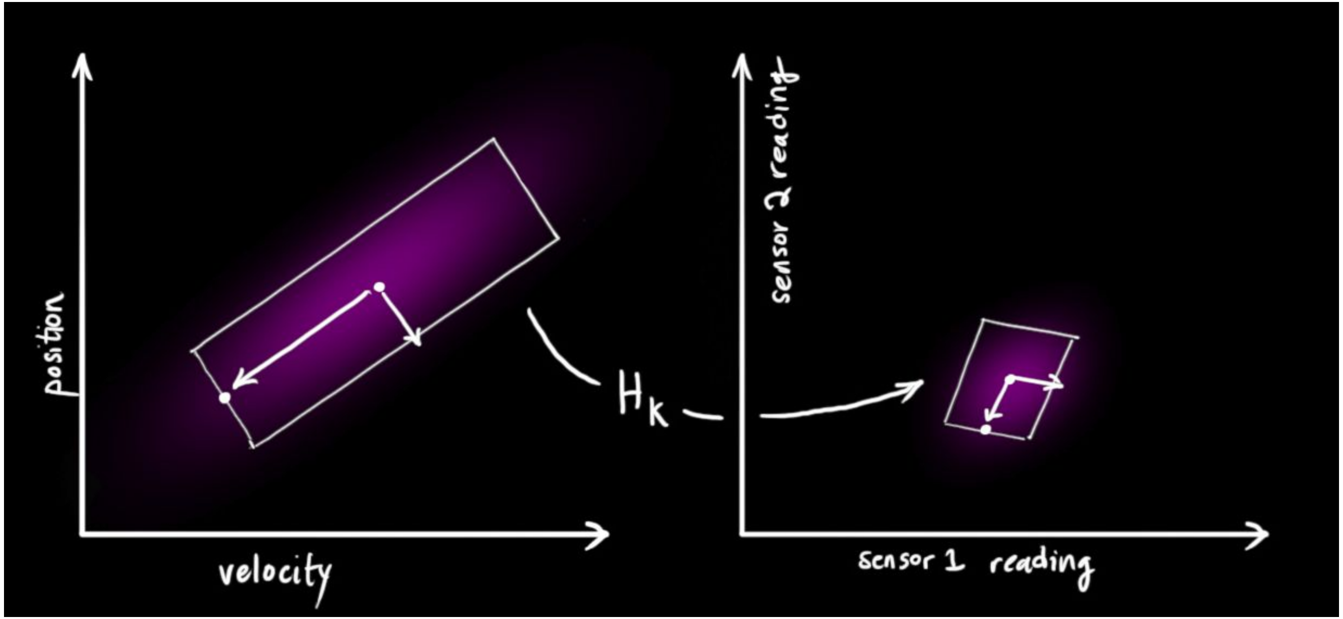
Source: https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/

Kalman Filter: Measurement



Source: https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/

Kalman Filter: Measurement



Source: https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/

Kalman Filter: Measurement

Measurement and state vectors may have different size, because not all state vector elements may be observable.

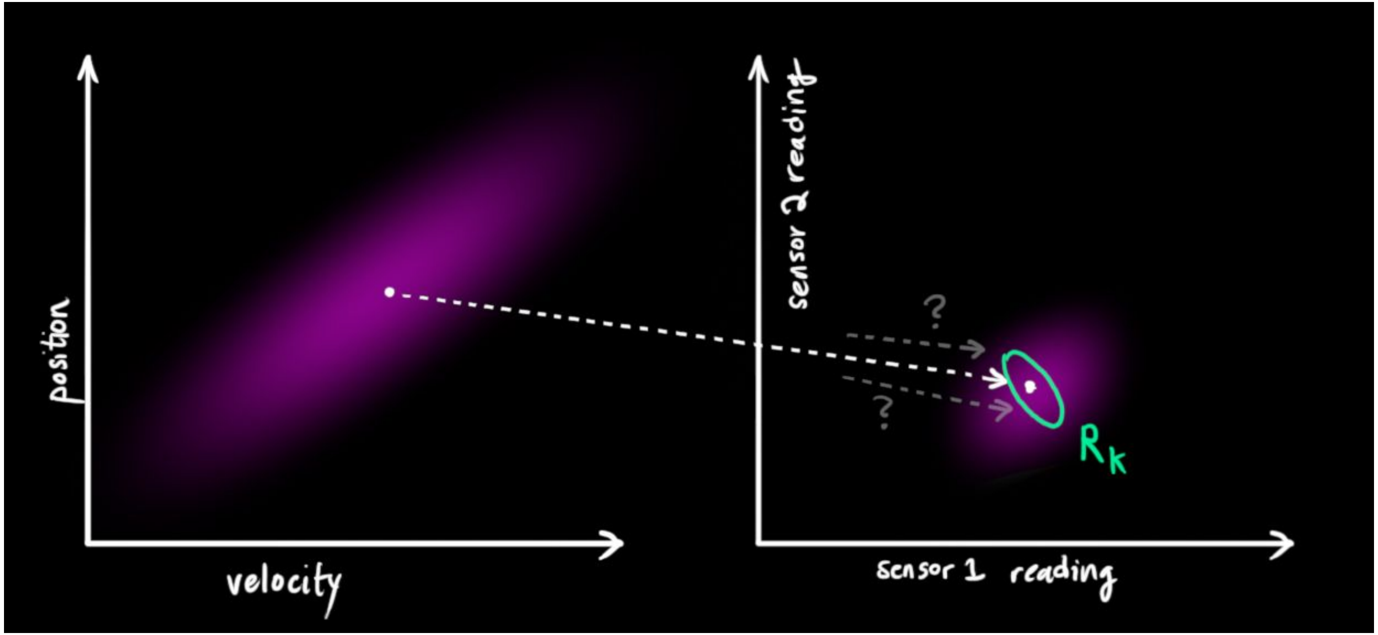
In trivial case, H is diagonal.

$$\vec{\mu}_{\text{expected}} = \mathbf{H}_k \hat{\mathbf{x}}_k$$

$$\mathbf{\Sigma}_{\text{expected}} = \mathbf{H}_k \mathbf{P}_k \mathbf{H}_k^T$$

Source: https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/

Kalman Filter: Measurement



Source: https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/

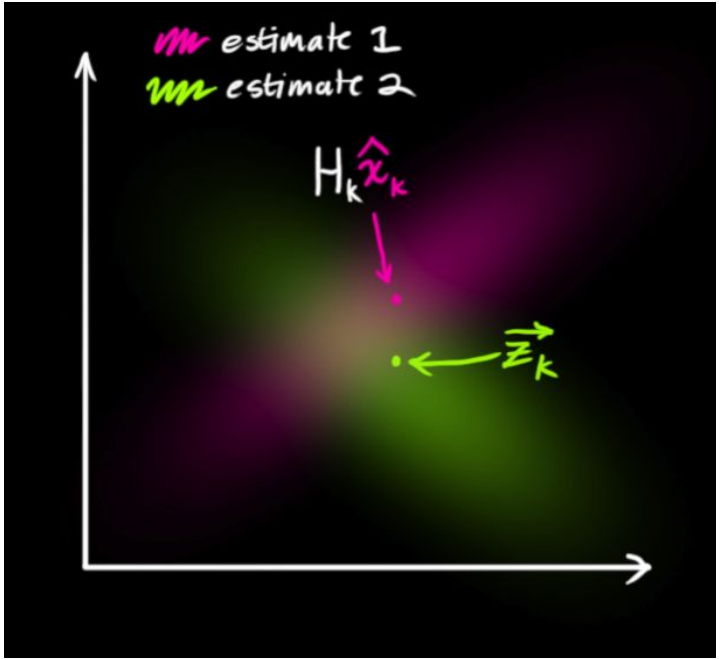
Kalman Filter: Combining estimates

We have two estimates:

1) Predicted by the model
2) Measured

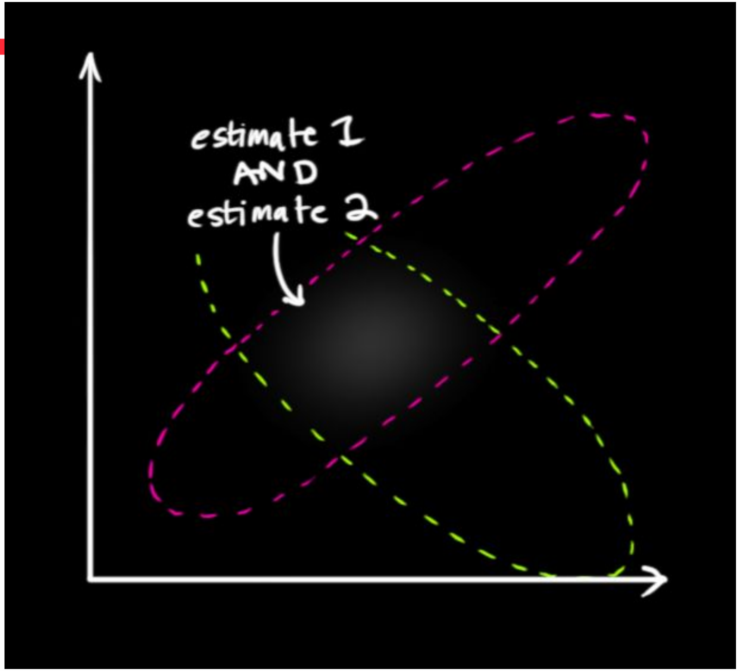
Both are gaussians.

How do we combine them?



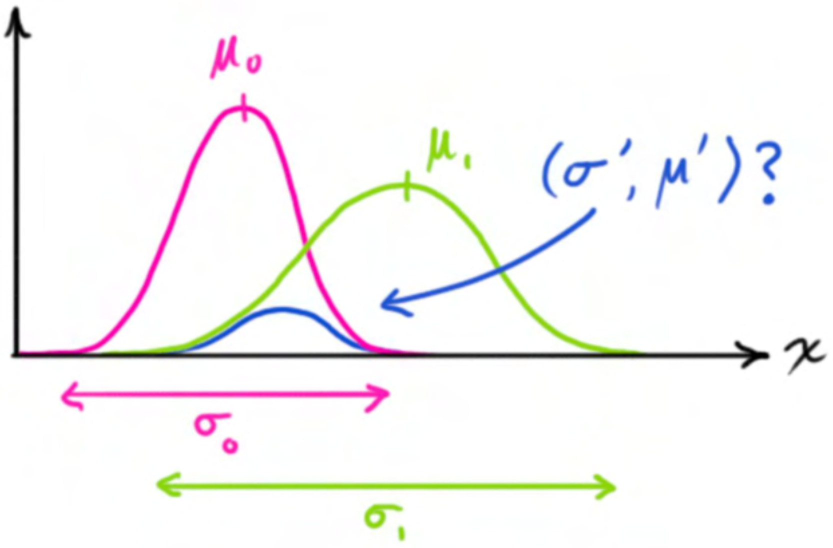
Source: https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/

Kalman Filter: Combining estimates



Source: https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/

Kalman Filter: Combining estimates



$$\mathcal{N}(x, \mu_0, \sigma_0) \cdot \mathcal{N}(x, \mu_1, \sigma_1) \stackrel{?}{=} \mathcal{N}(x, \mu', \sigma')$$

Source: https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/

Kalman Filter: Combining estimates

$$\mathbf{K} = \Sigma_0(\Sigma_0 + \Sigma_1)^{-1}$$

$$\vec{\mu}' = \vec{\mu_0} + \mathbf{K}(\vec{\mu_1} - \vec{\mu_0})$$
$$\Sigma' = \Sigma_0 - \mathbf{K}\Sigma_0$$

K - Kalman Gain

Source: https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/

Kalman Filter: Update

The predicted measurement with $(\mu_0, \Sigma_0) = (\mathbf{H}_k \hat{\mathbf{x}}_k, \mathbf{H}_k \mathbf{P}_k \mathbf{H}_k^T)$,

and the observed measurement with $(\mu_1, \Sigma_1) = (\vec{\mathbf{z}_k}, \mathbf{R}_k)$.

$$\hat{\mathbf{x}}'_k = \hat{\mathbf{x}}_k + \mathbf{K}'(\vec{\mathbf{z}_k} - \mathbf{H}_k \hat{\mathbf{x}}_k)$$
$$\mathbf{P}'_k = \mathbf{P}_k - \mathbf{K}' \mathbf{H}_k \mathbf{P}_k$$

$$\mathbf{K}' = \mathbf{P}_k \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k \mathbf{H}_k^T + \mathbf{R}_k)^{-1}$$

- **Kalman Gain**

Source: https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/

Kalman Filter

Kalman Filter hyperparameters:

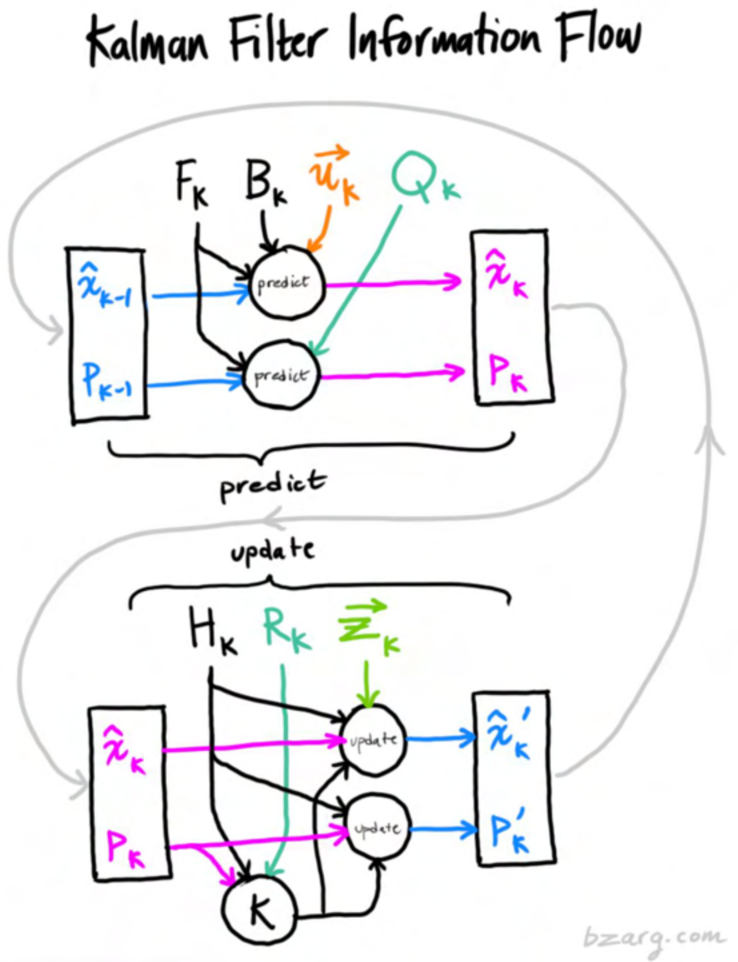
F - state transition matrix.

H - measurement matrix.

P - initial state uncertainty covariance matrix

Q - model noise covariance (model uncertainty).

R - measurement noise covariance (measurement uncertainty).



Source: https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/

Kalman Filter: Questions

Kalman Filter: Assignment

Notebook: seminar1-kalman.ipynb

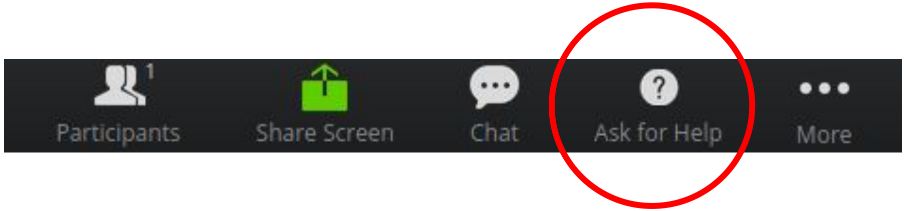
Zip archive in Discord

Work in groups

40 minutes

Ask your teammates

Call the lecturer if you need help



SORT: Questions

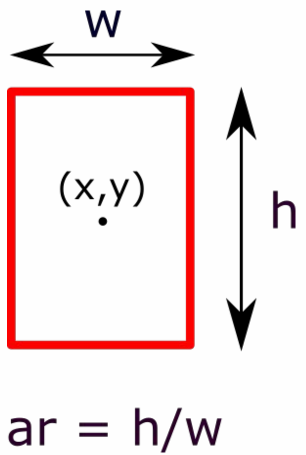
Kalman Filter: Questions

SORT

SORT: states and measurements

Each tracklet = Kalman Filter with state [x, y, ar, h, x’, y’, h’]:

1. Bbox center on x axis.
2. Bbox center on y axis.
3. Aspect ratio.
4. Height.
5. Movement speed along x.
6. Movement speed along y.
7. Height change speed.



Detections are used as measurements for Kalman update: state vector size is 7, measurement vector size is 4 (only [x, y, ar, h]).

SORT: assignment

How to associate new detections to tracks?

SORT: assignment

How to associate new detections to tracks? **- by IoU**.

Linear assignment problem on matrix:

CostMatrix = Tracks x Detections.

Hungarian algorithm is used.

SORT

Minimum IoU threshold of predicted bbox and tracked bbox is required.

If IoU for some detection is less than threshold => new track is initiated.

SORT: parameters

We keep “age” of each track (frame count).

1) **max_age** - How many frames to wait until track is deleted.
2) **min_hits** - Minimal track age required to keep the track.
3) **iou_thresh** - Minimal IoU threshold to match detection with a track.

If track is finished before **min_hits** => it is ignored (detector False Positive).

Track states:

1) Initiated: found unmatched detection
2) Confirmed : age > **min_hits**
3) Missed : no detection at this timestep
4) Deleted : time since last update > **max_age**

SORT: usage, tips, applications

1) Works very fast on CPU.
2) Minimum FPS is required: about 5 FPS for faces/pedestrians.
3) Tune tracker hyperparameters (previous slide).

Applications:

1) Face/Object Recognition:
   a) Compute centroid embedding for “good” subset of crops in a track => AKNN => propagate label on full track. Much better embedding quality.
   b) Keep only 1 centroid for each track in database. 1 embedding instead of embeddings of all objects in a track => faster AKNN and smaller base.
   c) Video Analytics.

SORT: Questions

SORT: Assignment

Notebook: seminar2-sort.ipynb

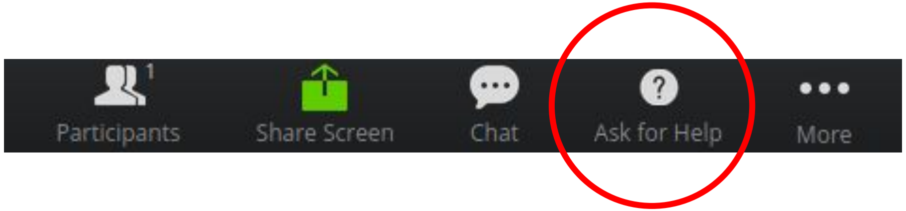
Zip archive in Discord

Work in groups

30 minutes

Ask your teammates

Call the lecturer if you need help



DeepSORT

1) Applicable if we have object descriptors (embeddings).
2) Associate detections with tracklets by embedding similarity.
3) Unmatched detections are associated by IoU (covered faces/pedestrians).
4) Matching cascade.
5) Gating mechanism.

DeepSORT: Matching cascade

Keep a collection of embeddings from N previous frames.

At timestep T, try to assign detections to tracks with hungarian algorithm. Start with embeddings from step T-1, T-2, etc..

Use MSE/Cosine Distance between embeddings to compute CostMatrix.

DeepSORT: Matching cascade

Keep a collection of embeddings from N previous frames.

At timestep T, try to assign detections to tracks with hungarian algorithm. Start with embeddings from step T-1, T-2, etc..

Use MSE/Cosine Distance between embeddings to compute CostMatrix.

In the end, try to match all unmatched detections with IoU CostMatrix.

Problem: We do not use any spatial information about tracks and bboxes. => use gating mechanism.

DeepSORT: Gating

Modify CostMatrix with gating values:

If distance between i-th measurement and j-th state (it’s projection) is greater than threshold => $CostMatrix_{ij}$ is replaced with infinity/bigger value.

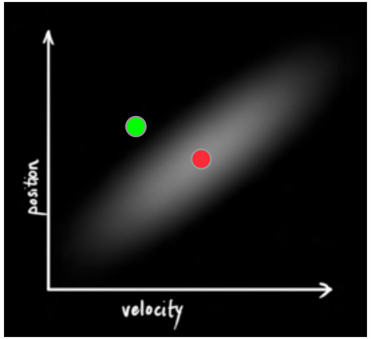
Track state is a normal **distribution** => we can measure distance from point to mean, but it’s bad.

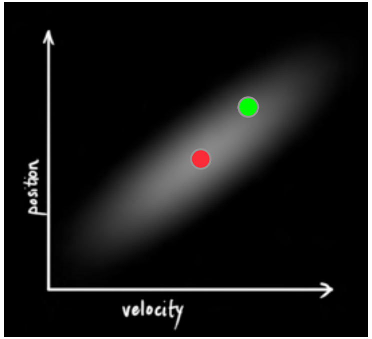
Problem: how to measure a distance from point to distribution? => mahalanobis distance

DeepSORT: Gating

Modify CostMatrix with gating values:

If mahalanobis distance between i-th measurement and j-th state (it's projection) is greater than threshold => $CostMatrix_{ij}$ is replaced with infinity/bigger value.





DeepSORT: Mahalanobis Distance

Squared Mahalanobis distance: $$d^{(1)}(i,j) = (\boldsymbol{d}_j - \boldsymbol{y}_i)^{\mathrm{T}} \boldsymbol{S}_i{}^{-1} (\boldsymbol{d}_j - \boldsymbol{y}_i),$$

y_i , S_i - i-th track state projection into measurement space

d_j - j-th detection.

“Distance from point to distribution”.

Relation to MLE:

$\ln(L) = - d^2 - \text{const}$

DeepSORT: Overview

At timestep T:

- Update kalman states.
- For t in [T-1, T-2, ..., T-N]:
  - Compute CostMatrix between unmatched embeddings and embeddings at timestep t.
  - Apply gating to CostMatrix.
  - Apply hungarian to match some of the detections.
- Match the remaining detections with IoU cost matrix (hungarian algorithm).
- Initiate new tracks, delete old ones.

Deep SORT: Questions

Deep SORT: Assignment

Notebook: seminar3-deep-sort.ipynb

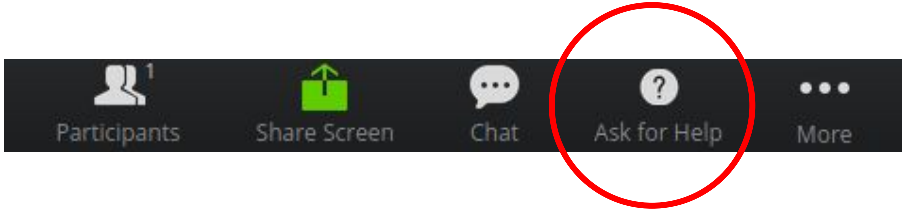
Zip archive in Discord

Work in groups

20 minutes

Ask your teammates

Call the lecturer if you need help



Deep SORT: Questions

Metrics

- FAF(↓): number of false alarms per frame.
- MT(↑): number of mostly tracked trajectories. I.e. target has the same label for at least 80% of its life span.
- ML(↓): number of mostly lost trajectories. i.e. target is not tracked for at least 20% of its life span.
- FP(↓): number of false detections.
- FN(↓): number of missed detections.
- ID sw(↓): number of times an ID switches to a different previously tracked object [24].
- Frag(↓): number of fragmentations where a track is interrupted by miss detection.

Metrics

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

- Multiple Object Tracking Accuracy.

$$\text{MOTP} = \frac{\sum_{t,i} d_{t,i}}{\sum_t c_t}$$

- Multiple Object Tracking Precision. Average overlap between all correctly matched hypotheses and their respective objects.

Datasets and benchmarks

Dataset	Classes	Videos Eval.	Videos Train	Avg length (s)	Tracks / video	Min resolution	Ann. fps	Total Eval length (s)
MOT17 [42]	1	7	7	35.4	112	640x480	30	248
KITTI [25]	2	29	21	12.6	52	1242x375	10	365
UA-DETRAC [64]	4	40	60	56	57.6	960x540	5	2,240
ImageNet-Vid [52]	30	1,314	4,000	10.6	2.4	480x270	~25	13,928
YTVIS [70]	40	645	2,238	4.6	1.7	320x240	5	2,967
TAO (Ours)	833	2,407	500	36.8	5.9	640x480	1	88,605

MOT challenge: https://motchallenge.net/

Source: https://arxiv.org/pdf/2005.10356.pdf

More trackers

SMOT: Single-Shot Multi Object Tracking:

https://arxiv.org/pdf/2010.16031v1.pdf

Fast Online Object Tracking and Segmentation: A Unifying Approach:

https://arxiv.org/pdf/1812.05050v2.pdf

FairMOT: On the Fairness of Detection and Re-Identification in Multiple Object Tracking

https://arxiv.org/pdf/2004.01888v5.pdf