Investigation of Adaptive Filtering Techniques for Vehicle Tracking in Videos

LINDSAY WHITE

LKWHITE@ENG.UCSD.EDU

Background & Applications

- Goal: Predict the motion of and track multiple vehicles in surveillance-style traffic videos

Applications:

- Vehicle counting [Tamersoy et al. 2010]
- Speed/traffic flow analysis on roadways [Morris et al. 2008]
- Detect collision [Mukhtar et al. 2015]
- Driver behavior analysis [Sivaraman et al. 2013]
  - Lane changes
  - Turns at intersections

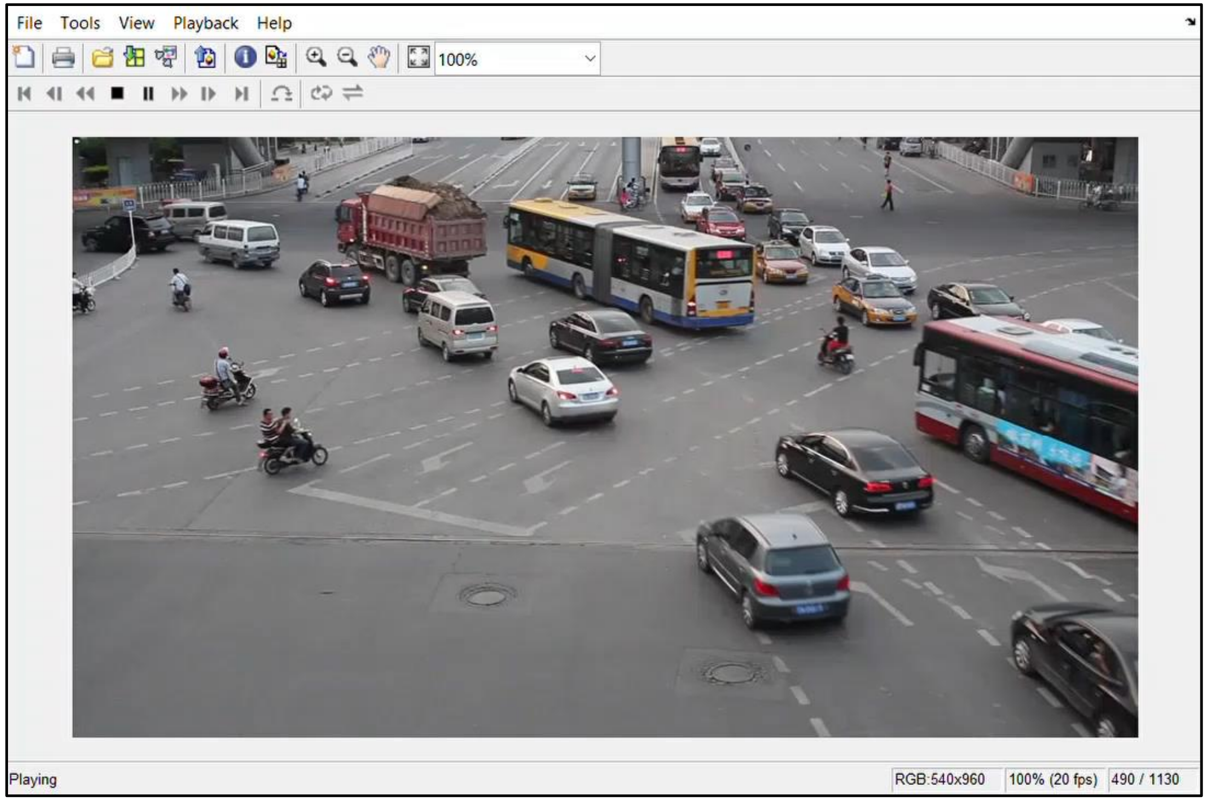


Image Source: UA-DETRAC: [Wen et al. 2020], [Lyu et al. 2018], [Lyu et al. 2017]

Problem Challenges & Proposed Solutions

- Detecting the vehicles in each frame
  - This is usually solved with a trained machine learning model trained on vehicles [Zhang et al. 2020]
  - We will use ground-truth centroid locations as the “detected” centroid location for each vehicle in each frame
- Motion model for the vehicles
  - This determines the setup for the prediction filters
    - Kalman Filter
    - Unscented Kalman Filter
  - We will use kinematics equations
- Associating vehicle detections to tracks
  - Challenge is distinguishing and tracking the motion of vehicles between frames
  - Munkres’ Assignment Algorithm

Prediction and Tracking Algorithm

All three of the cars in this frame have a centroid location and a filter associated with them

This frame is in the middle of the video, not at initialization

The centroid is in red

Base Image Source: UA-DETRAC: [Wen et al. 2020], [Lyu et al. 2018], [Lyu et al. 2017]

Prediction and Tracking Algorithm

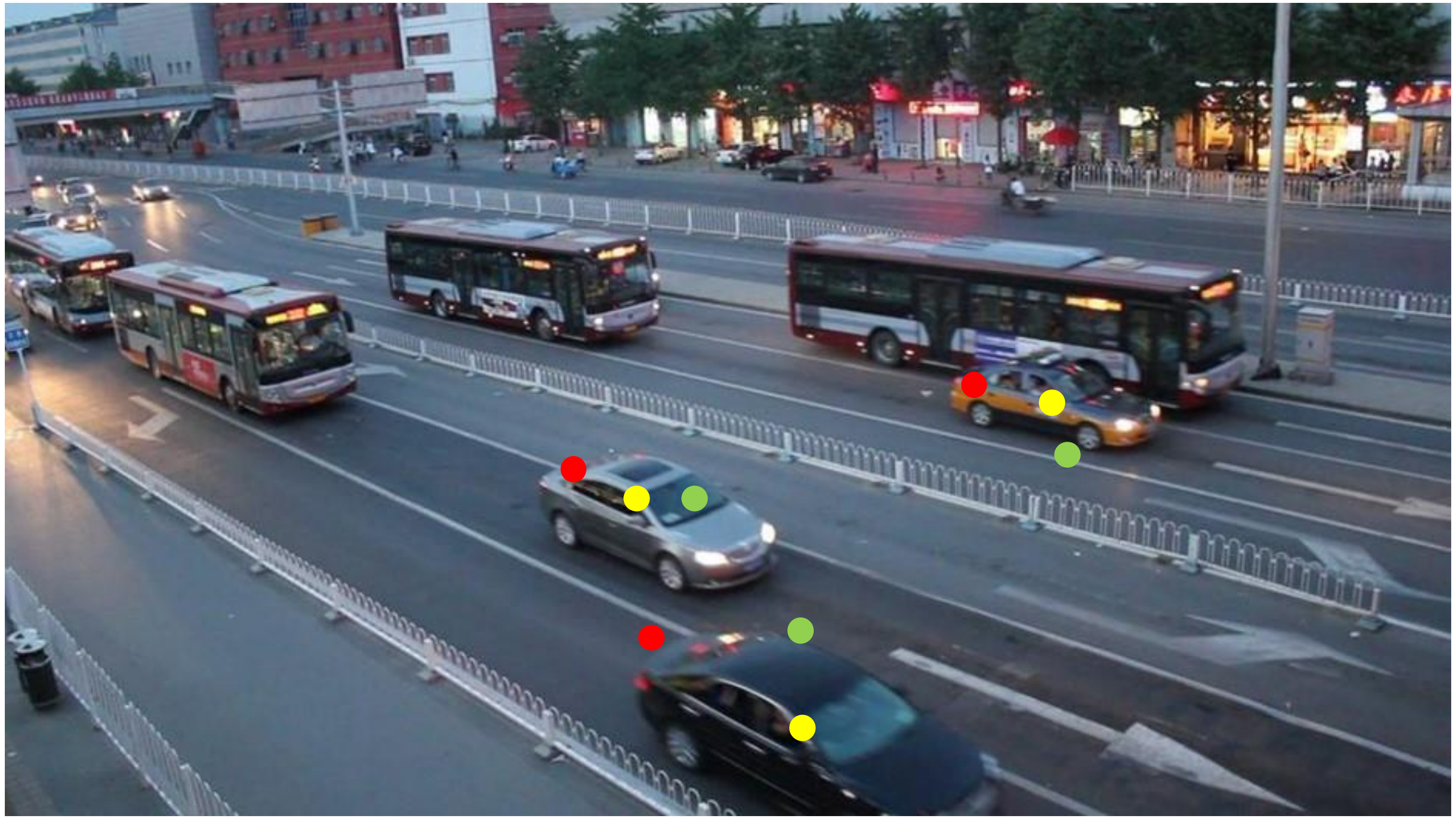
1. Each filter outputs its predicted centroid location for the next frame

The centroid is in red
The track prediction is in green

Base Image Source: UA-DETRAC: [Wen et al. 2020], [Lyu et al. 2018], [Lyu et al. 2017]

Prediction and Tracking Algorithm

2. Move to the next frame and detect vehicles

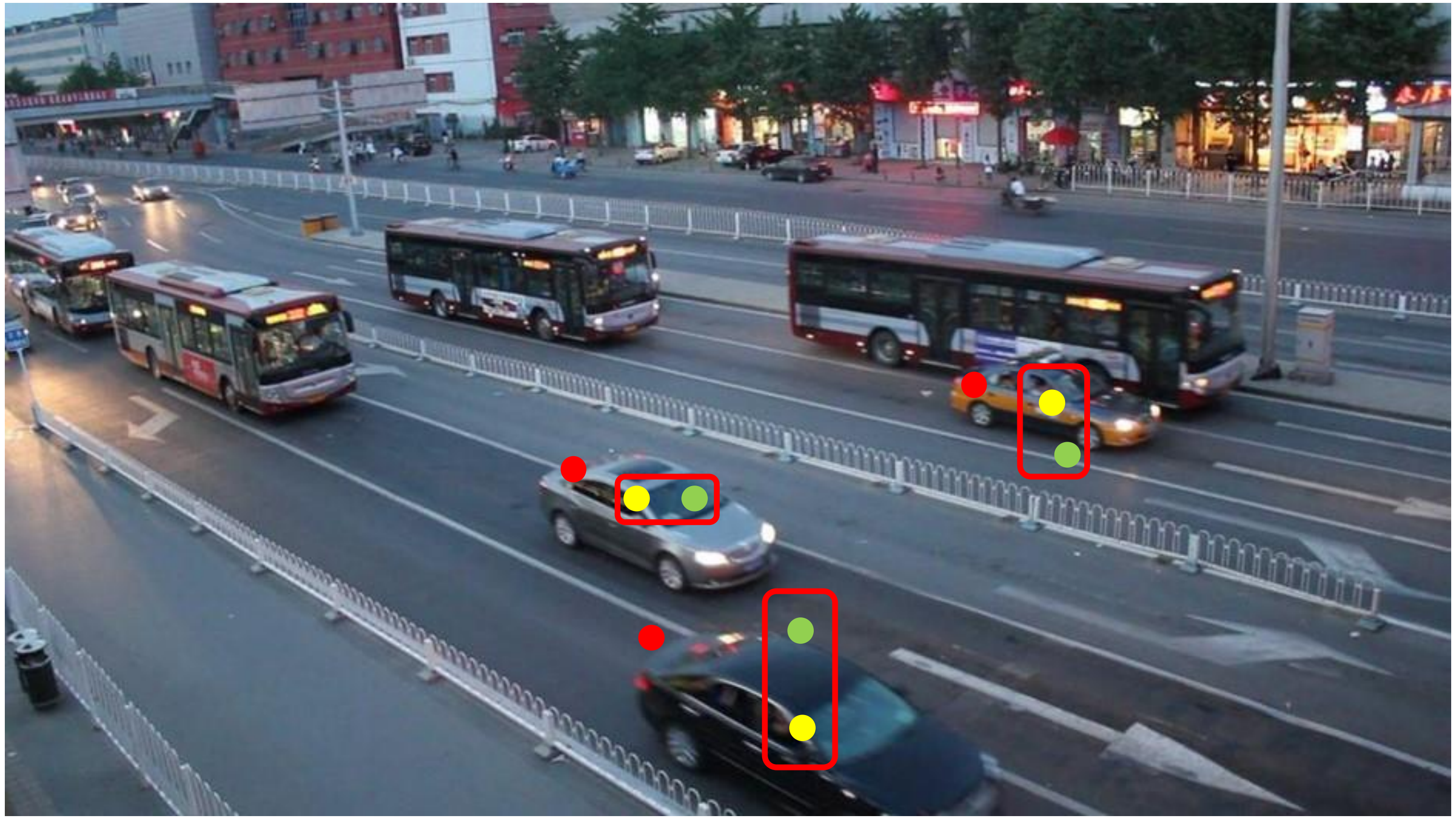


The centroid is in red
The track prediction is in green
The detection is in yellow

Base Image Source: UA-DETRAC: [Wen et al. 2020], [Lyu et al. 2018], [Lyu et al. 2017]

Prediction and Tracking Algorithm

3. Use Munkres' Assignment Algorithm to associate tracks and detections

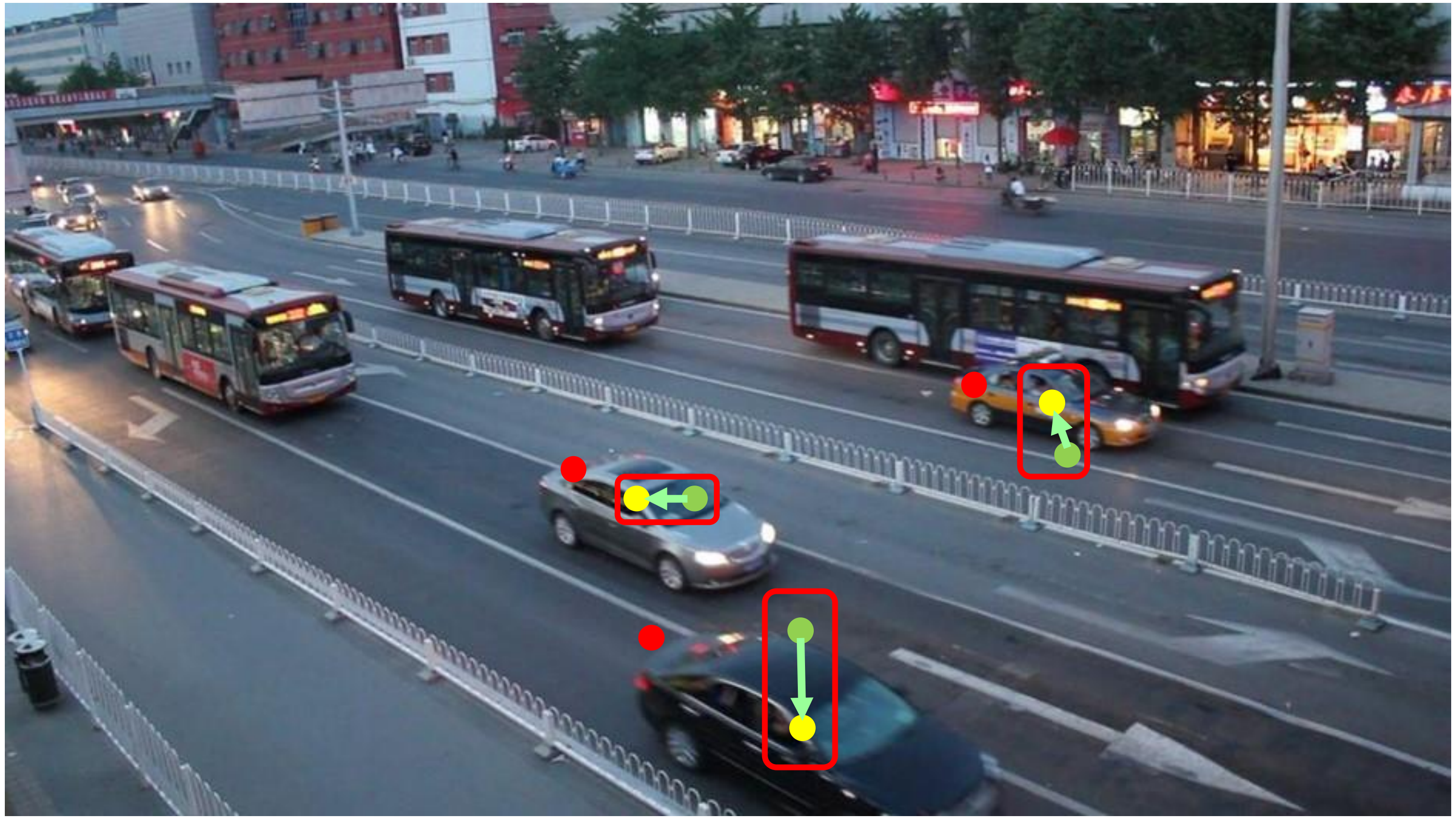


The centroid is in red
The track prediction is in green
The detection is in yellow

Prediction and Tracking Algorithm

4. Calculate the Euclidean distance in pixels between the estimated centroid and the detected centroid

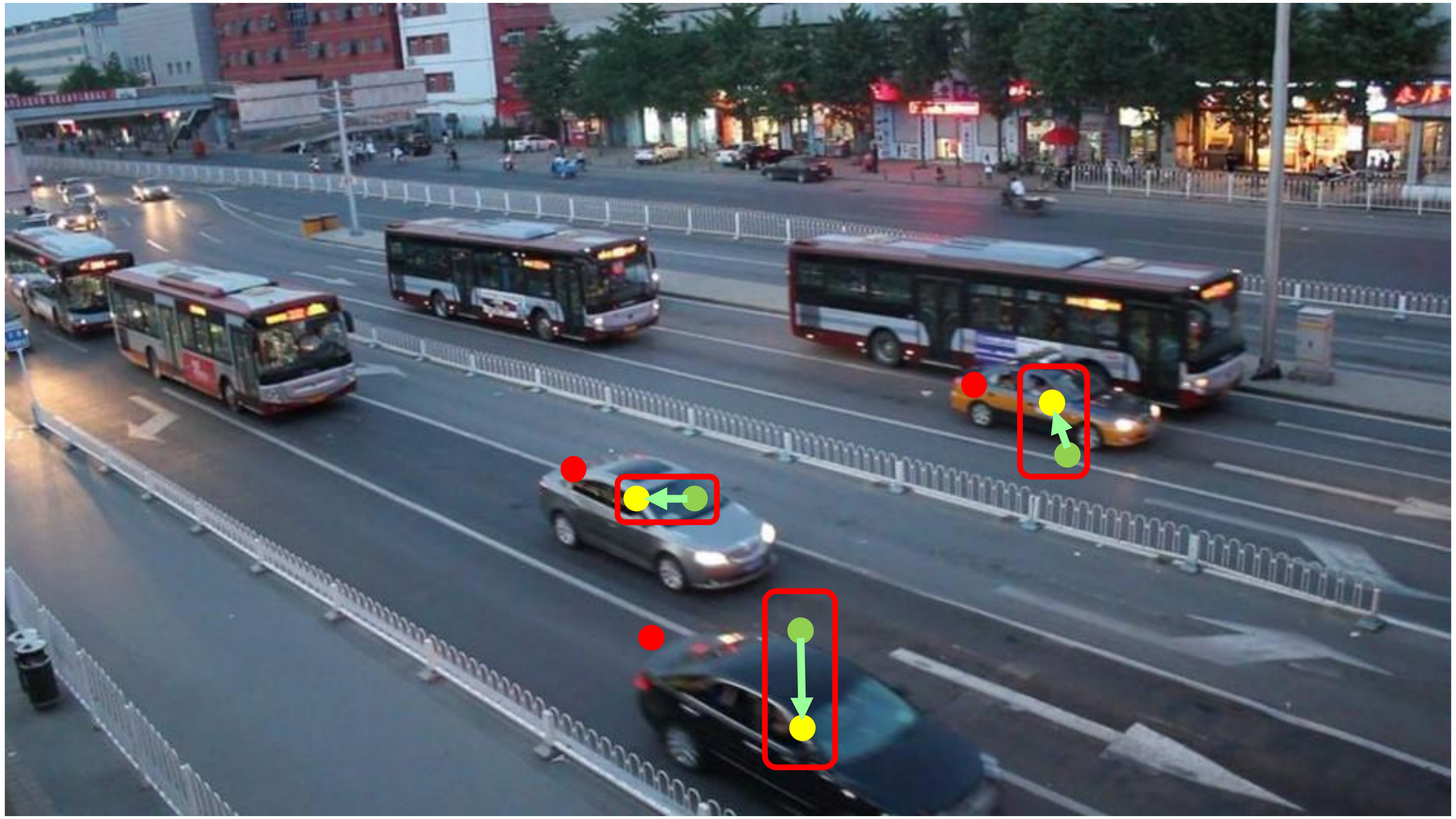
This is the method of comparison between filters and during parameter tuning



The centroid is in red
The track prediction is in green
The detection is in yellow

Prediction and Tracking Algorithm

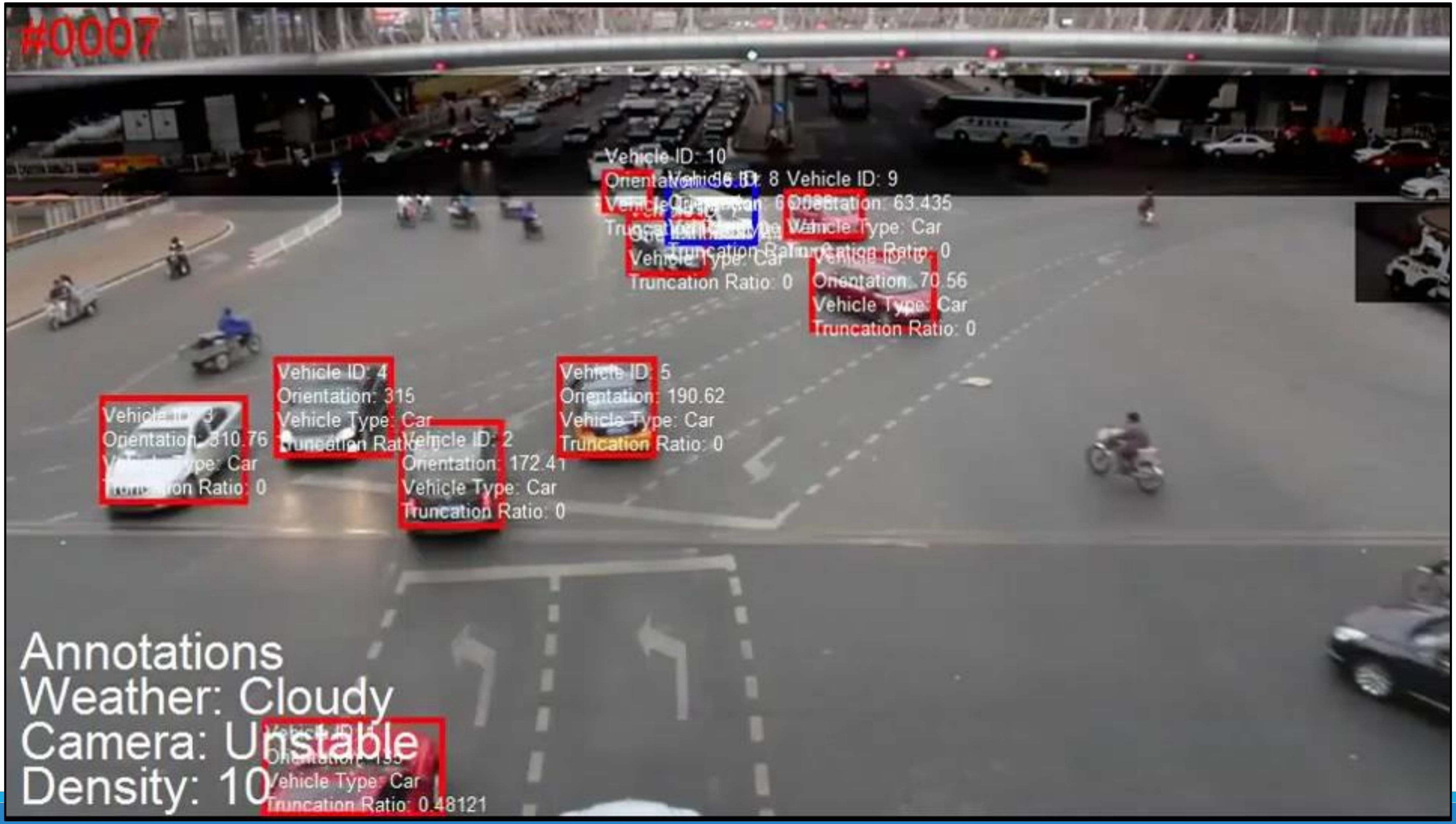
5. Use the assigned detected location to calculate the *a posteriori* state estimate



The centroid is in red
The track prediction is in green
The detection is in yellow

University at Albany DEtection and TRACking (UA-DETRAC) Dataset

[Wen et al. 2020]
[Lyu et al. 2018]
[Lyu et al. 2017]



Video Source: https://detrac-db.rit.albany.edu/

Munkres’ Assignment Algorithm

- Algorithm to solve an optimization problem for associating pairs [Munkres 1957]
- Generate a cost matrix between all potential pairs, then determine the overall pairings that give the lowest total cost
- In our specific problem, we want to associate tracks with the most likely detected centroid in a given frame
  - The cost value is the Euclidean distance between the detected centroids and the predicted centroid from the filter
- In a given frame, there are N detections and M tracks with predictions for new locations, with a cost matrix given by:
  - Each detection is a column, each row is a track

$$C = \begin{array}{cc} \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1N} \\ d_{21} & d_{22} & \dots & d_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ d_{M1} & d_{M2} & \dots & d_{MN} \end{bmatrix} & \downarrow \text{Tracks} \\ \xrightarrow{\text{Detections}} & \end{array}$$

Munkres' Assignment Algorithm

Let N = M = 4

[Munkres 1957]

$$C = \begin{bmatrix} d_{11} & d_{12} & d_{13} & d_{14} \\ d_{21} & d_{22} & d_{23} & d_{24} \\ d_{31} & d_{32} & d_{33} & d_{34} \\ d_{41} & d_{42} & d_{43} & d_{44} \end{bmatrix}$$

Tracks

Detections

- Given the cost matrix, subtract the smallest cost from each element of each row
  - This results in at least one zero per row
  - Assuming there is only one zero per column, the zeros indicate the detection-to-track pairings
- In this example, map (row to column):
  - Track 1 to Detection 4
  - Track 2 to Detection 2
  - Track 3 to Detection 1
  - Track 4 to Detection 3

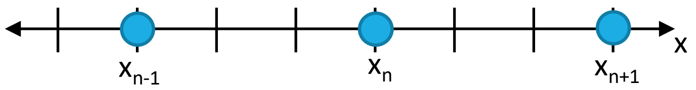
$$C' = \begin{bmatrix} d'_{11} & d'_{12} & d'_{13} & 0 \\ d'_{21} & 0 & d'_{23} & d'_{24} \\ 0 & d'_{32} & d'_{33} & d'_{34} \\ d'_{41} & d'_{42} & 0 & d'_{44} \end{bmatrix}$$

From Lecture 16:
State update equation: $\boldsymbol{x}(n+1) = \boldsymbol{F}(n+1,n)\boldsymbol{x}(n) + \boldsymbol{v}_1(n)$
$\boldsymbol{v}_1(n)$ is system noise
Observation: $\boldsymbol{y}(n) = \boldsymbol{C}(n)\boldsymbol{x}(n) + \boldsymbol{v}_2(n)$
$\boldsymbol{v}_2(n)$ is measurement noise

Kalman Filter

- Constant-Velocity Kinematics Equation:
  - $x_n = x_{n-1} + v_x \Delta t$

$v_x = 3$
$\Delta t = 1$



- State Model: $\mathbf{s} = \begin{bmatrix} x \\ v_x \\ y \\ v_y \end{bmatrix}$

In class, the state model is denoted **x**
To avoid confusion with the x-coordinate,
it has been named **s** for "state" here

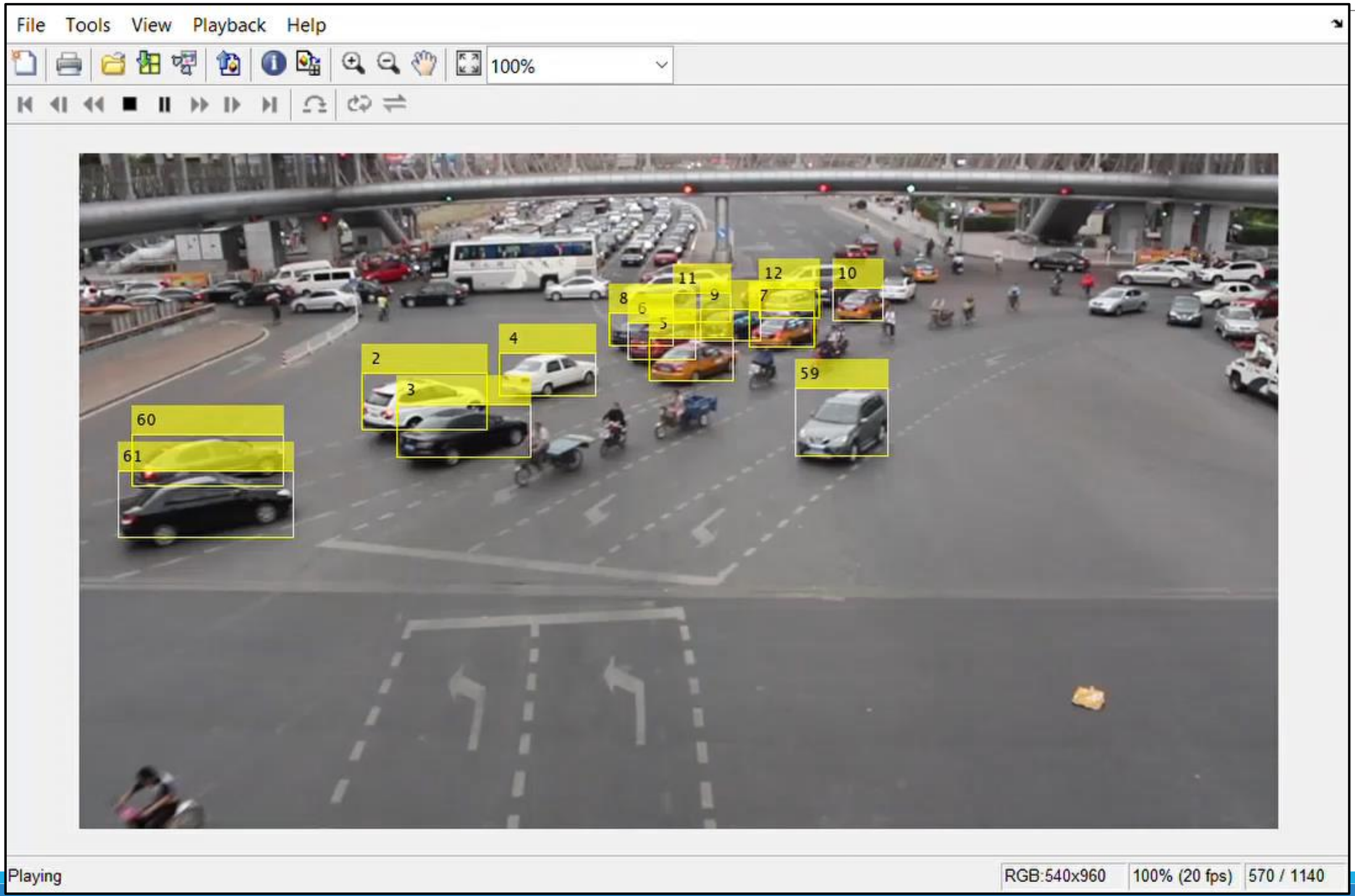
- State Transition Matrix: $\boldsymbol{F} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

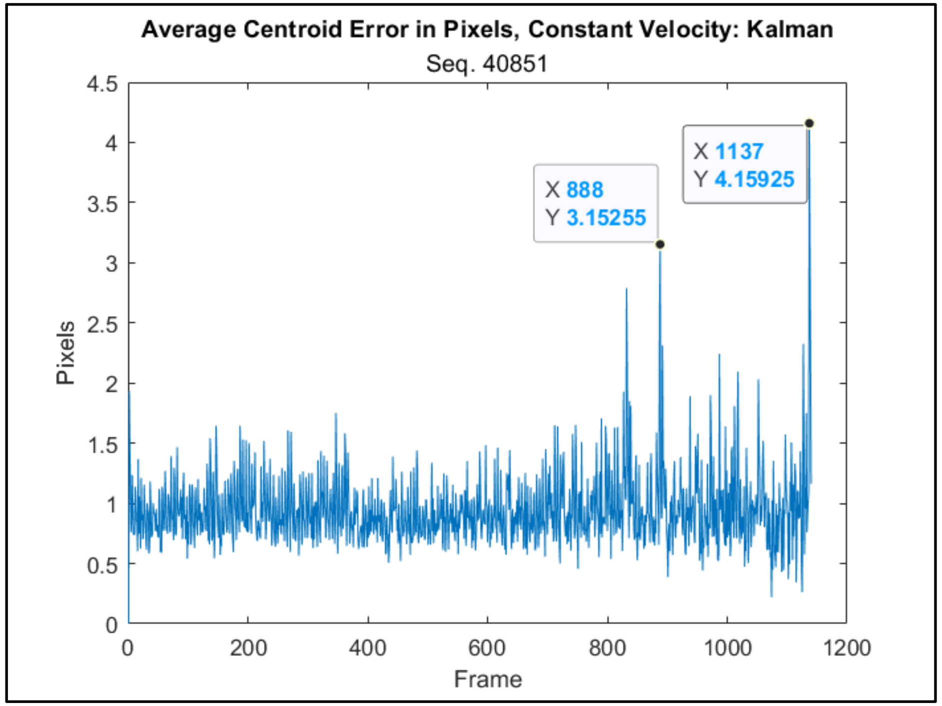
- Measurement Matrix: $\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

Update Equation:

$$\boldsymbol{s}_{n+1} = \begin{bmatrix} x(n+1) \\ v_x(n+1) \\ y(n+1) \\ v_y(n+1) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x(n) \\ v_x(n) \\ y(n) \\ v_y(n) \end{bmatrix}$$

Kalman Filter





Results after parameter tuning

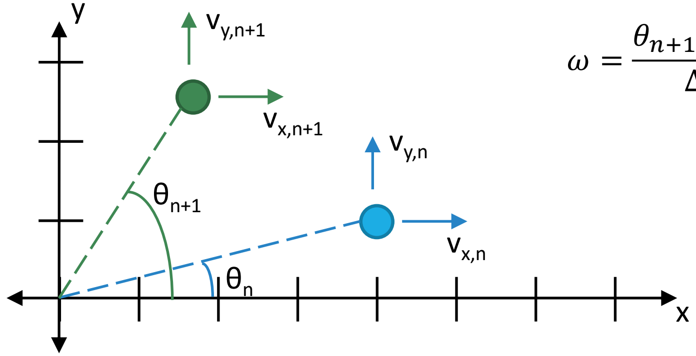
Unscented Kalman Filter: Constant Turn

[Yuan et al. 2014]
[Julier and Uhlmann 1997]

Constant Velocity:

$$\mathbf{s} = \begin{bmatrix} x \\ v_x \\ y \\ v_y \end{bmatrix}$$

ω is the turn rate



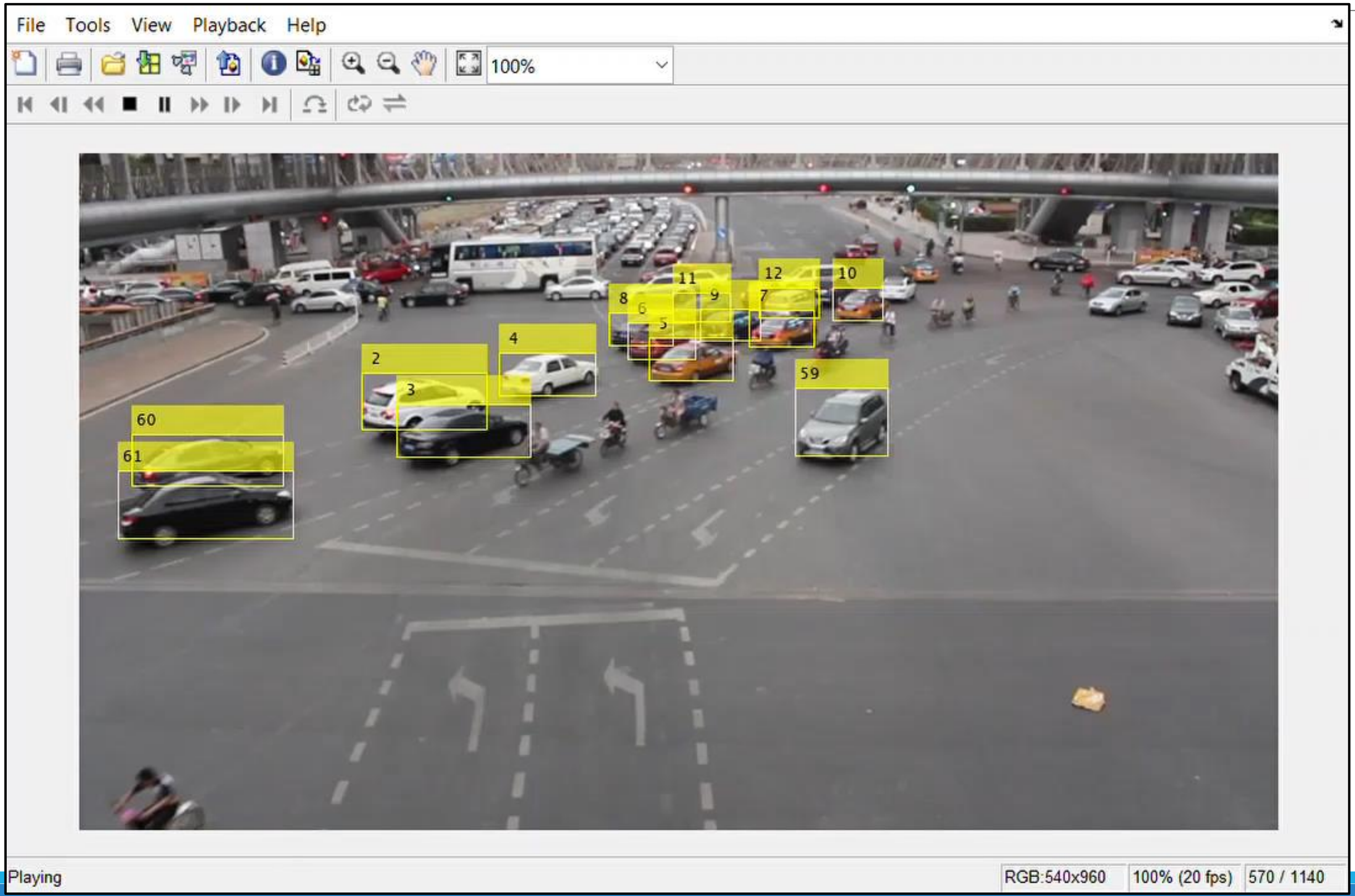
$$\omega = \frac{\theta_{n+1} - \theta_n}{\Delta t}$$

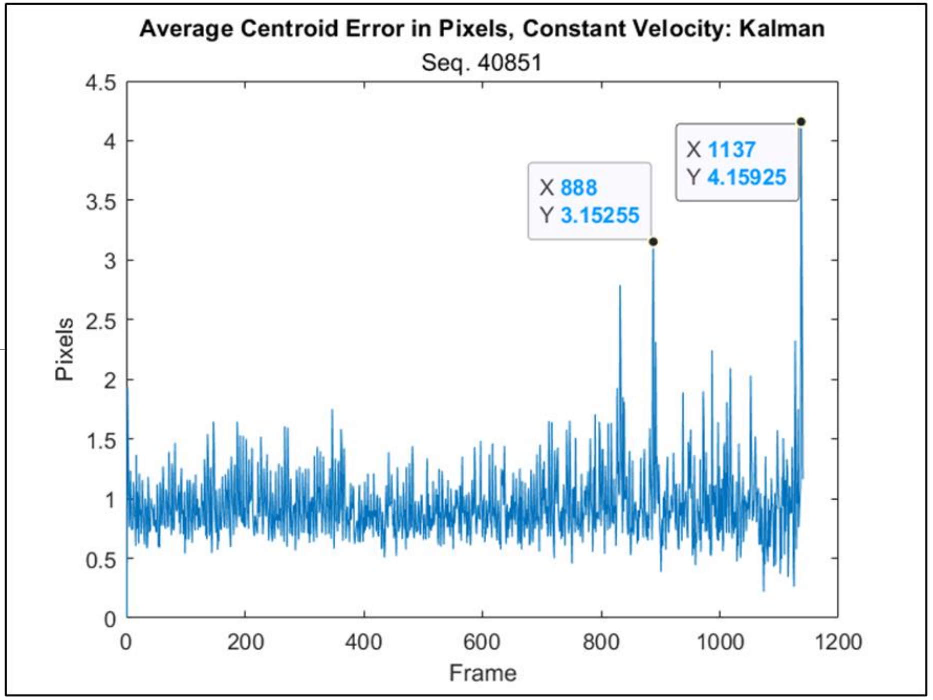
- State Model: $\mathbf{s} = \begin{bmatrix} x \\ v_x \\ y \\ v_y \\ \omega \end{bmatrix}$

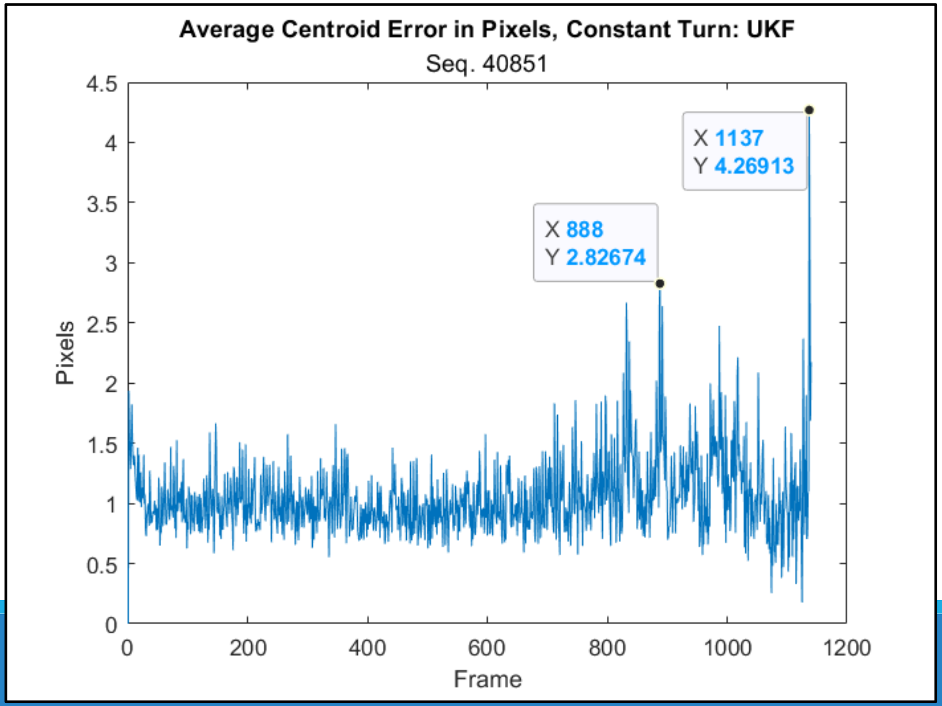
- State Transition Equation: $\boldsymbol{s}_{n+1} = \begin{bmatrix} x_{n+1} \\ v_{x,n+1} \\ y_{n+1} \\ v_{y,n+1} \\ \omega_{n+1} \end{bmatrix} = \begin{bmatrix} x_n + \frac{v_{x,n}}{\omega_n}\sin(\omega_n \Delta t) - \frac{v_{y,n}}{\omega_n}(1 - cos(\omega_n \Delta t)) \\ v_{x,n}\cos(\omega_n \Delta t) - v_{y,n}\sin(\omega_n \Delta t) \\ x_n + \frac{v_{x,n}}{\omega_n}(1 - cos(\omega_n \Delta t)) + \frac{v_{y,n}}{\omega_n}\sin(\omega_n \Delta t) \\ v_{x,n}\sin(\omega_n \Delta t) + v_{y,n}\cos(\omega_n \Delta t) \\ \omega_n \end{bmatrix}$

- Measurement Matrix: $\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$

Unscented Kalman Filter







Results after parameter tuning

Conclusions

- Goal:
  - Investigate adaptive filtering techniques for predicting and tracking vehicular motion in surveillance-style videos
- Results:
  - The linear Kalman filter with the constant velocity model performs very well on this data set
  - The nonlinear Unscented Kalman filter with the constant turn model also performs very well, but slightly worse on average than the constant velocity model
- Next Steps:
  - Complete parameter tuning with the Unscented Kalman Filter
  - Consider other motion models for the Unscented Kalman Filter
  - Investigate the Particle Filter

References

DeTrac

1. L. Wen, D. Du, Z. Cai, Z. Lei, M. Chang, H. Qi, J. Lim, M. Yang, and S. Lyu, “UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking,” *Computer Vision and Image Understanding*, 2020.
2. S. Lyu, M.-C. Chang, D. Du, W. Li, Y. Wei, M. Del Coco, P. Carcagn`ı, A. Schumann, B. Munjal, D.-H. Choi et al., “UA-DETRAC 2018: Report of AVSS2018 & IWT4S challenge on advanced traffic monitoring,” in *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2018, pp. 1–6.
3. S. Lyu, M.-C. Chang, D. Du, L. Wen, H. Qi, Y. Li, Y. Wei, L. Ke, T. Hu, M. Del Coco et al., “UA-DETRAC 2017: Report of avss2017 & IWT4S challenge on advanced traffic monitoring,” in *Advanced Video and Signal Based Surveillance (AVSS), 2017 14th IEEE International Conference on. IEEE*, 2017, pp. 1–7.

Tracking and Modeling

1. J. Munkres, “Algorithms for the assignment and transportation problems,” *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957. [Online]. Available: http://www.jstor.org/stable/2098689
2. N. L. Baisa, “Derivationa of a constant velocity motion model for visual tracking,” *CoRR*, vol. abs/2005.00844, 2020. [Online]. Available: https://arxiv.org/abs/2005.00844
3. X. Yuan, F. Lian, and C. Han, “Models and algorithms for tracking target with coordinated turn motion, ”*Mathematical Problems in Engineering*, vol. 2014, pp. 1–10, 2014. [Online]. Available: https://doi.org/10.1155/2014/649276

References

Applications

1. B. Tamersoy and J. Aggarwal, “Counting vehicles in highway surveil-lance videos,” in *2010 20th International Conference on Pattern Recognition*, 2010, pp. 3631–3635.

2. B. T. Morris and M. M. Trivedi, “Learning, modeling, and classification of vehicle track patterns from live video,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 3, pp. 425–437, 2008.

3. A. Mukhtar, L. Xia, and T. B. Tang, “Vehicle detection techniques for collision avoidance systems: A review,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2318–2338, 2015.

4. S. Sivaraman and M. M. Trivedi, “Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1773–1795, 2013.

5. K. Zhang, H. Ren, Y. Wei, and J. Gong, “Multi-target vehicle detection and tracking based on video,” in *2020 Chinese Control And Decision Conference* (CCDC), 2020, pp. 3317–3322.

Filters

1. M. Vasconcelos, Class Lecture, Topic: ”Kalman Filter.” ECE251B, University of California, San Diego, La Jolla, California, May 20, 2021.

2. S. J. Julier and J. K. Uhlmann, “A New Extension of the Kalman Filter to Nonlinear Systems,” In *Proc. of AeroSense: The 11th Int. Symp. On Aerospace/Defense Sensing, Simulation and Controls*, 1997.

3. S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," in *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401-422, March 2004, doi: 10.1109/JPROC.2003.823141.

Bonus Slides

Q&A BACKUP VISUALS

Related Problems

- Our problem: Vehicle prediction and tracking in surveillance-style videos
  - Specifically, we are interested only in solutions involving adaptive filtering
  - Modern solutions are moving towards machine learning in general [Zhang et al. 2020]
- Vehicle detection in images
  - Typical approaches involve machine learning [Zhang et al. 2020]
- Vehicle detection, prediction, and tracking in onboard vehicle videos
  - Typical approaches for detection are also machine learning
  - Prediction and tracking must take dynamics of the “host” vehicle into consideration as well as the measurements and estimated dynamics of the detected vehicles
  - Usually also involves machine learning

Method of Comparison

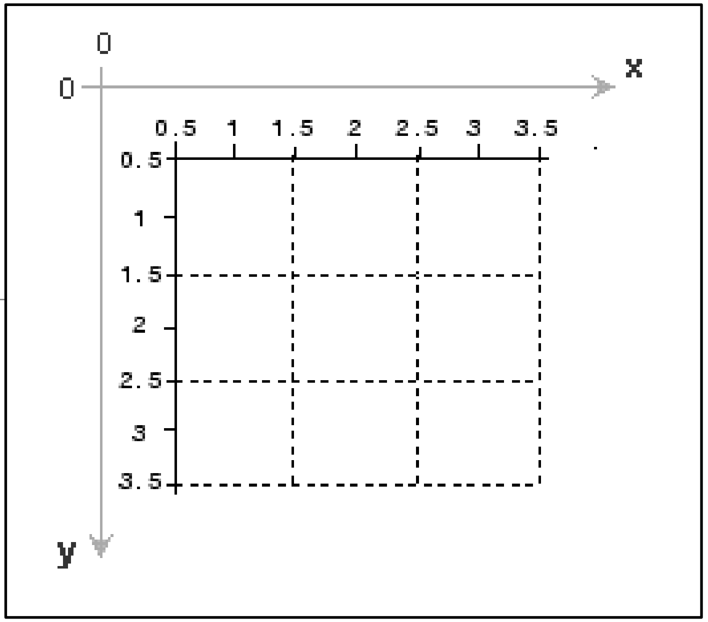


Image source:
https://www.mathworks.com/help/images/image-coordinate-systems.html

- MATLAB pixel coordinate system uses fractional values to represent points within any given pixel
  - Ground-truth centroid values all end in x.5 or x.0
  - Example: If the height in pixels is an odd number, expect to have the centroid ground truth value end in x.5
- Euclidean distance between predicted centroid point and ground-truth centroid point
  - Two points, $\mathrm{P} = (x_P, y_P)$ and $\mathrm{G} = (x_G, y_G)$
  - $dist = \sqrt{(y_P - y_G)^2 + (x_P - x_G)^2}$
- When the filter is doing a good job performing the prediction step, the Euclidean distance between the two center points is expected to be small

Kalman Filter Equations

- State update equation
  - $\boldsymbol{x}(n+1) = \boldsymbol{F}(n+1,n)\boldsymbol{x}(n) + \boldsymbol{v}_1(n)$
- *A Priori* mean estimate
  - $\hat{\boldsymbol{x}}(n|Y_{n-1}) = \boldsymbol{F}(n,n-1)\hat{\boldsymbol{x}}(n-1|Y_{n-1})$
- *A Priori* covariance estimate
  - $\boldsymbol{K}(n,n-1) = \boldsymbol{F}(n,n-1)\boldsymbol{K}(n-1)\boldsymbol{F}^H(n,n-1) + \boldsymbol{Q}_1(n-1)$
- Observation Prediction
  - $\hat{\boldsymbol{y}}(n|Y_{n-1}) = \boldsymbol{C}(n)\hat{\boldsymbol{x}}(n|Y_{n-1})$
- Innovation
  - $\boldsymbol{\alpha}(n) = \boldsymbol{y}(n) - \hat{\boldsymbol{y}}(n|Y_{n-1})$
- Innovation Covariance
  - $\boldsymbol{R}(n) = \boldsymbol{C}(n)\boldsymbol{K}(n,n-1)\boldsymbol{C}^H(n) + \boldsymbol{Q}_2(n)$
- Observation
  - $\boldsymbol{y}(n) = \boldsymbol{C}(n)\boldsymbol{x}(n) + \boldsymbol{v}_2(n)$
- Kalman Gain
  - $\boldsymbol{G}(n) = \boldsymbol{F}(n+1,n)\boldsymbol{K}(n,n-1)\boldsymbol{C}^H(n)\boldsymbol{R}^{-1}(n)$
- *A posteriori* state estimate
  - $\hat{\boldsymbol{x}}(n|Y_n) = \hat{\boldsymbol{x}}(n|Y_{n-1}) + \boldsymbol{G}(n)\boldsymbol{\alpha}(n)$
- *A posteriori* covariance
  - $\boldsymbol{K}(n) = \left(\boldsymbol{I} - \boldsymbol{F}(n+1,n)\boldsymbol{G}(n)\boldsymbol{C}(n)\right)\boldsymbol{K}(n,n-1)$

All equations taken from class lecture 16

Unscented Kalman Filter Equations

- Transformed state vector
  - $\hat{\boldsymbol{x}}(n) = \boldsymbol{F}\big(\boldsymbol{x}(n-1)\big)$
- *A Priori* mean estimate
  - $\hat{\boldsymbol{x}}(n|Y_{n-1}) = \boldsymbol{\mu}_{\hat{\boldsymbol{x}}} = \sum_{i=0}^{n} \boldsymbol{W}_i\, \hat{\boldsymbol{x}}(n)$
- *A Priori* covariance estimate
  - $\boldsymbol{K}(n, n-1) = \sum_{i=0}^{n} \boldsymbol{W}_i (\hat{\boldsymbol{x}}(n) - \boldsymbol{\mu}_{\hat{\boldsymbol{x}}})\, (\hat{\boldsymbol{x}}(n) - \boldsymbol{\mu}_{\hat{\boldsymbol{x}}})^{\boldsymbol{T}} + \boldsymbol{Q}_1(n-1)$
- Transformed observation prediction
  - $\hat{\boldsymbol{y}}(n|Y_{n-1}) = \boldsymbol{C}\big(\boldsymbol{x}(n-1)\big)$
- Innovation mean
  - $\boldsymbol{\alpha}(n) = \sum_{i=0}^{n} \boldsymbol{W}_i \hat{\boldsymbol{y}}(n|Y_{n-1})$
- Innovation Covariance
  - $\boldsymbol{R}(n) = \sum_{i=0}^{n} \boldsymbol{W}_i \big(\hat{\boldsymbol{y}}(n|Y_{n-1}) - \boldsymbol{\alpha}(n)\big)\big(\hat{\boldsymbol{y}}(n|Y_{n-1}) - \boldsymbol{\alpha}(n)\big)^T + \boldsymbol{Q}_2(n)$
- State and Innovation Covariance
  - $\boldsymbol{C}(n) = \sum_{i=0}^{n} \boldsymbol{W}_i (\hat{\boldsymbol{x}}(n) - \boldsymbol{\mu}_{\hat{\boldsymbol{x}}})\, \big(\hat{\boldsymbol{y}}(n|Y_{n-1}) - \boldsymbol{\alpha}(n)\big)^T$
- Gain
  - $\boldsymbol{G}(n) = \boldsymbol{C}(n)\boldsymbol{R}^{-1}(n)$
- *A posteriori* state estimate
  - $\boldsymbol{x}(n|Y_n) = \hat{\boldsymbol{x}}(n|Y_{n-1}) + \boldsymbol{K}(n)\big(\hat{\boldsymbol{y}}(n|Y_{n-1}) + \boldsymbol{v_2}(n) - \boldsymbol{\alpha}(n)\big)$
- *A posteriori* covariance
  - $\boldsymbol{K}(n) = \boldsymbol{K}(n, n-1) - \boldsymbol{G}_n \boldsymbol{R}(n) \boldsymbol{G}_n^T$

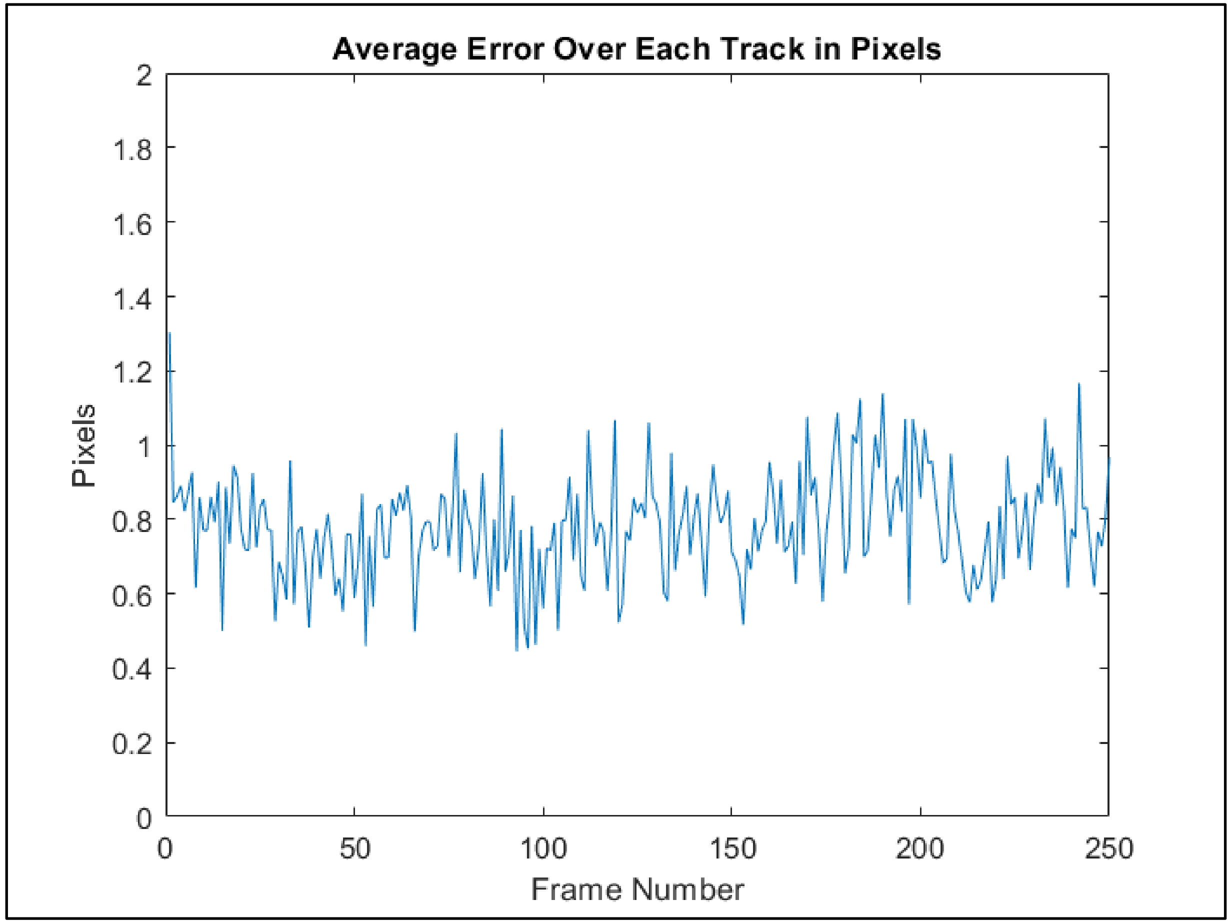
[Julier and Uhlmann 1997, 2004]

Unscented Kalman Filter

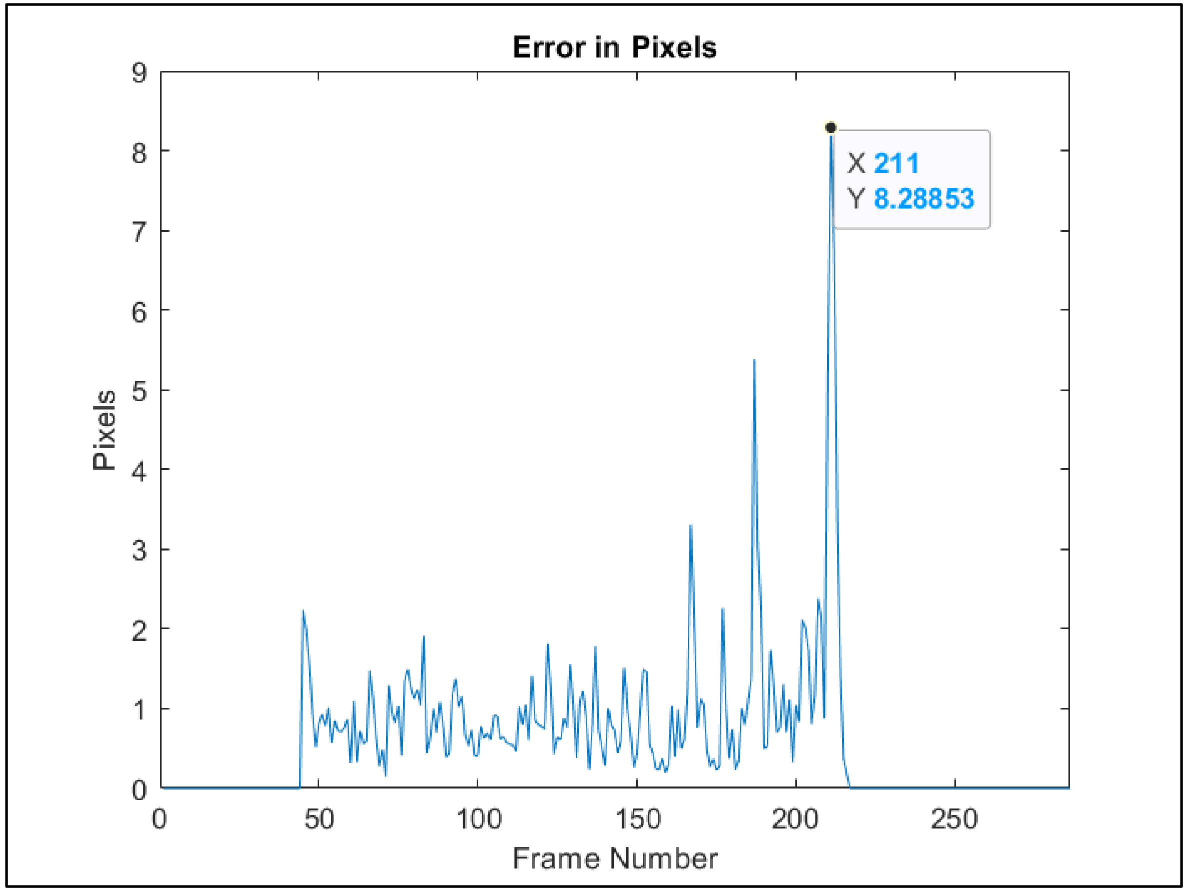
[Julier and Uhlmann 1997, 2004]

- Generate a set of n+1 “sigma points” that, taken together, have a Gaussian probability distribution with a given mean and a given covariance
  - Each point has a weight associated with it
- Apply the nonlinear filter function to each of the points, generating the transformed points
- Use the transformed points to calculate the transformed mean and covariance
- Estimate the transformed observation (innovation) mean and covariance by using the observation function and the weights of the sigma points
- Generate the innovation covariance matrix and a cross-covariance matrix between the innovation and transformed sigma points
- Perform the filter update with the standard Kalman filter equations from lecture

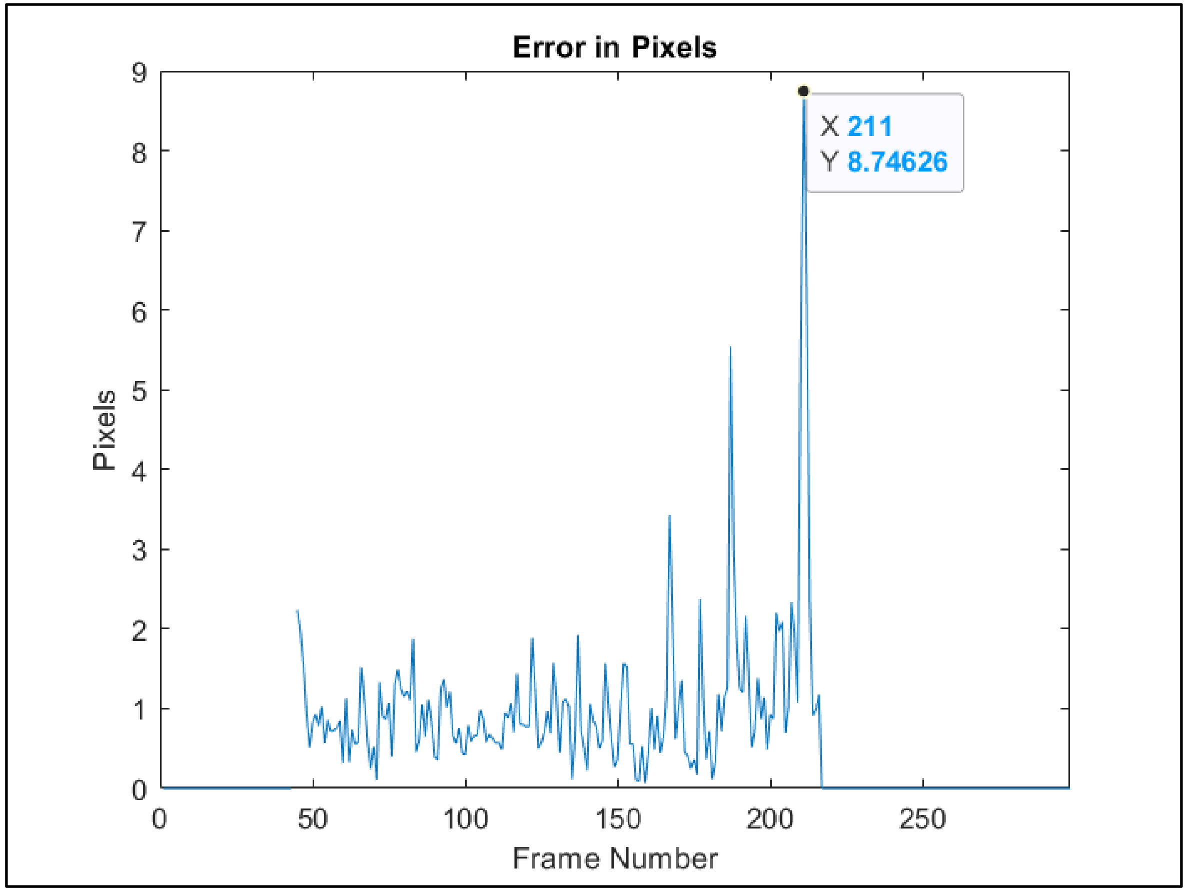
Measurement Error, $\boldsymbol{v}_2(n)$



Single Vehicle Pixel Error

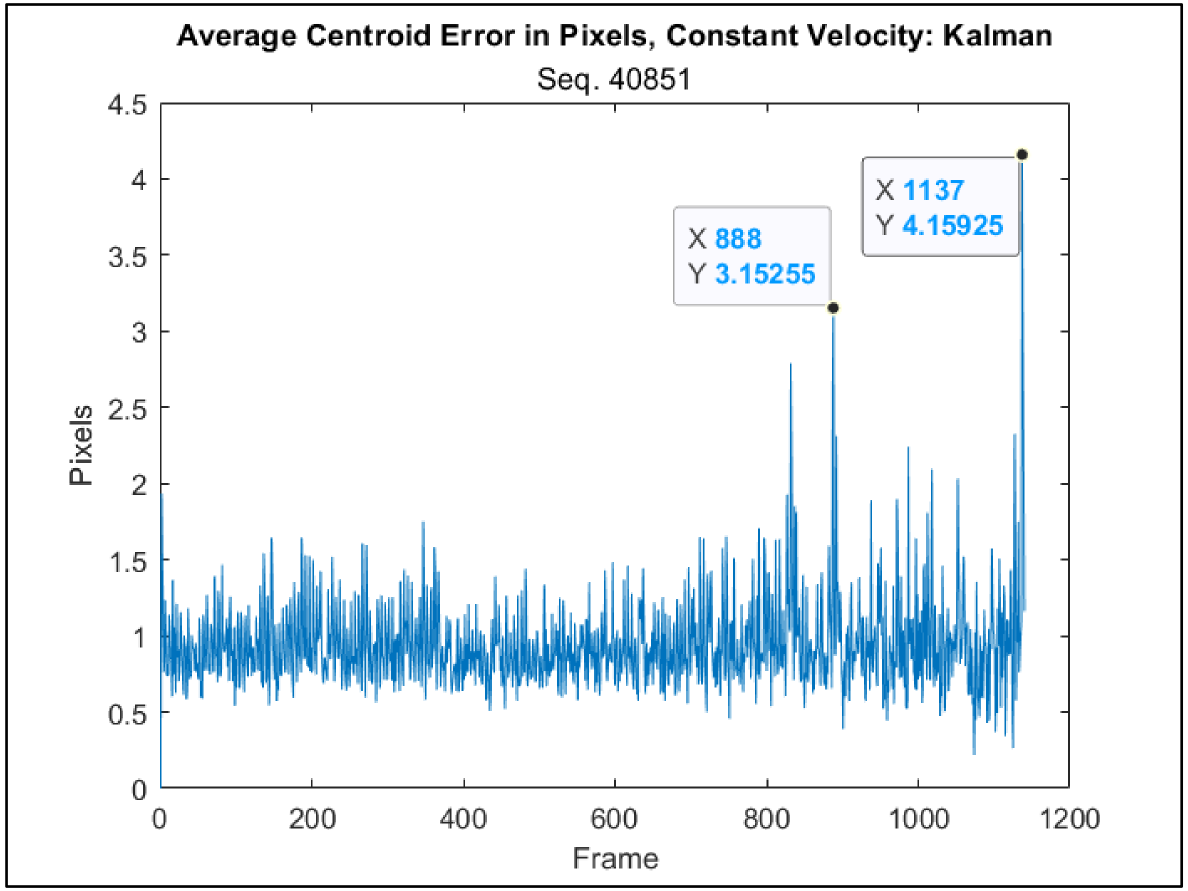


Kalman Filter

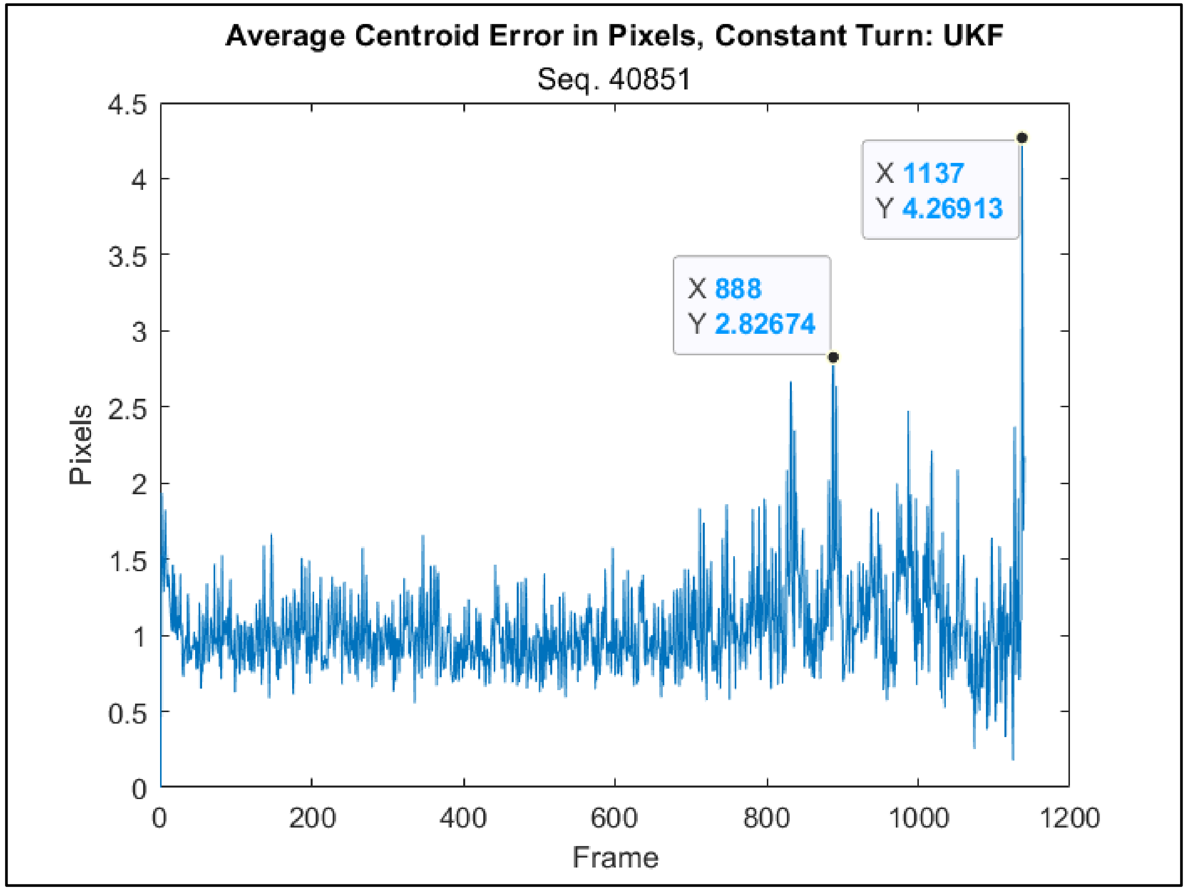


Unscented Kalman Filter

Multi-Car Average Pixel Error



Kalman Filter



Unscented Kalman Filter

Results after parameter tuning

Munkres' Assignment Algorithm

[Munkres 1957]

$$C' = \begin{bmatrix} d'_{11} & 0 & d'_{13} & 0 \\ d'_{21} & 0 & d'_{23} & d'_{24} \\ 0 & d'_{32} & d'_{33} & d'_{34} \\ 0 & d'_{42} & d'_{34} & d'_{44} \end{bmatrix}$$

- If there are no zeros in a given column, subtract the smallest value from all values of the column

$$C'' = \begin{bmatrix} d'_{11} & 0 & d''_{13} & 0 \\ d'_{21} & 0 & d''_{23} & d'_{24} \\ 0 & d'_{32} & d''_{33} & d'_{34} \\ 0 & d'_{42} & 0 & d'_{44} \end{bmatrix}$$

- If there are multiple zeros per row, either can be assigned and the other ignored
- In this example, map (row to column):
  - Track 1 to Detection 4
  - Track 2 to Detection 2
  - Track 3 to Detection 1
  - Track 4 to Detection 3

$$\begin{bmatrix} d'_{11} & \cancel{0} & d''_{13} & 0 \\ d'_{21} & 0 & d''_{23} & d'_{24} \\ 0 & d'_{32} & d''_{33} & d'_{34} \\ \cancel{0} & d'_{42} & 0 & d'_{44} \end{bmatrix}$$