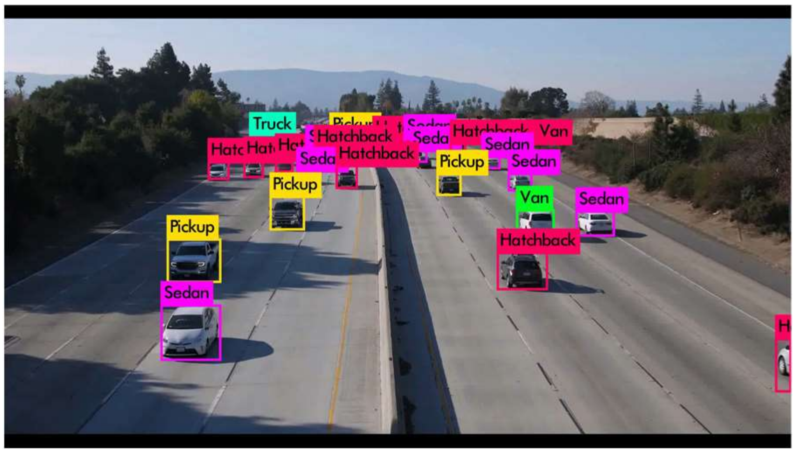
3D Object Tracking and Localization for AI City

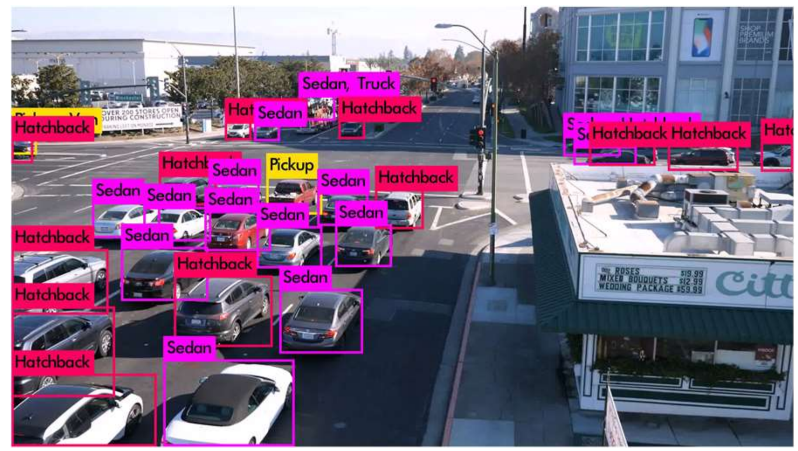
Gaoang Wang, Zheng Tang, Jenq-Neng Hwang

Information processing lab, University of Washington

UNIVERSITY *of* WASHINGTON

Success of CNN Vehicle Detectors (YOLOv2[1])



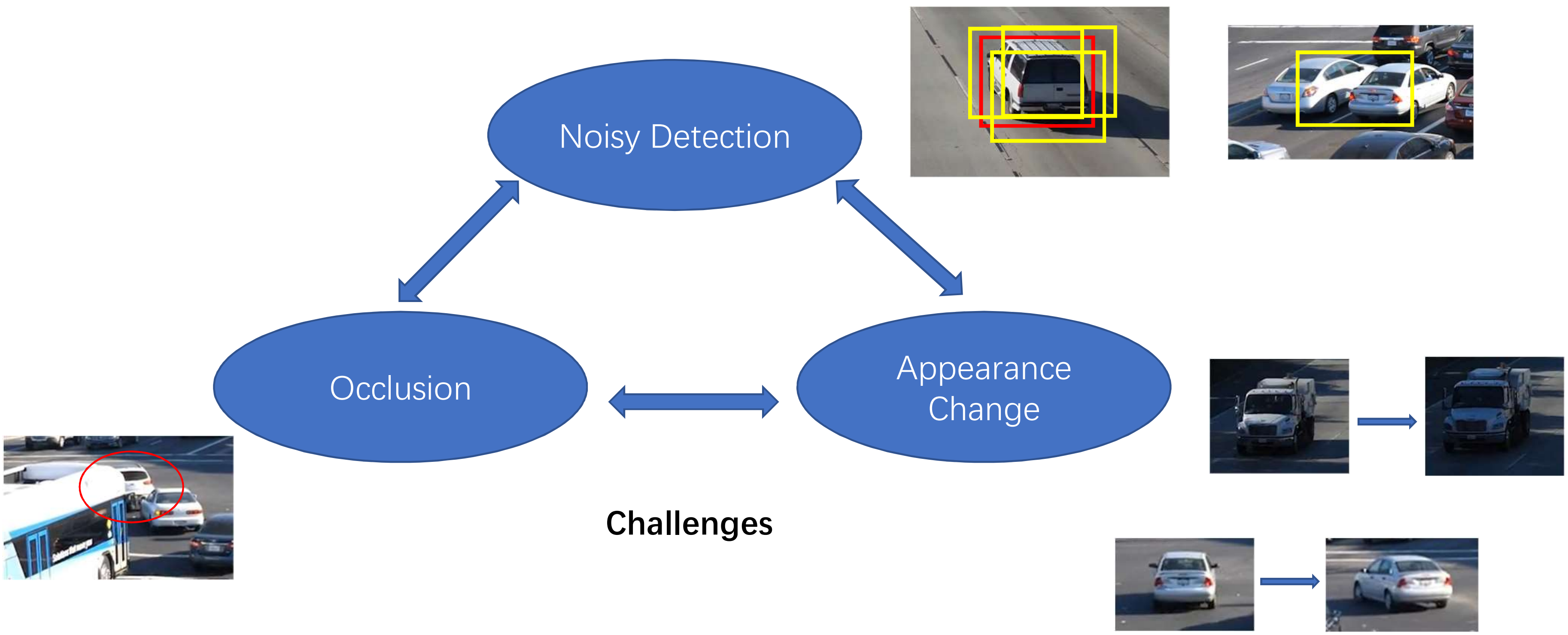


- Where are the cars in world coordinates?
- What is the GPS speed of each car?

3D object tracking

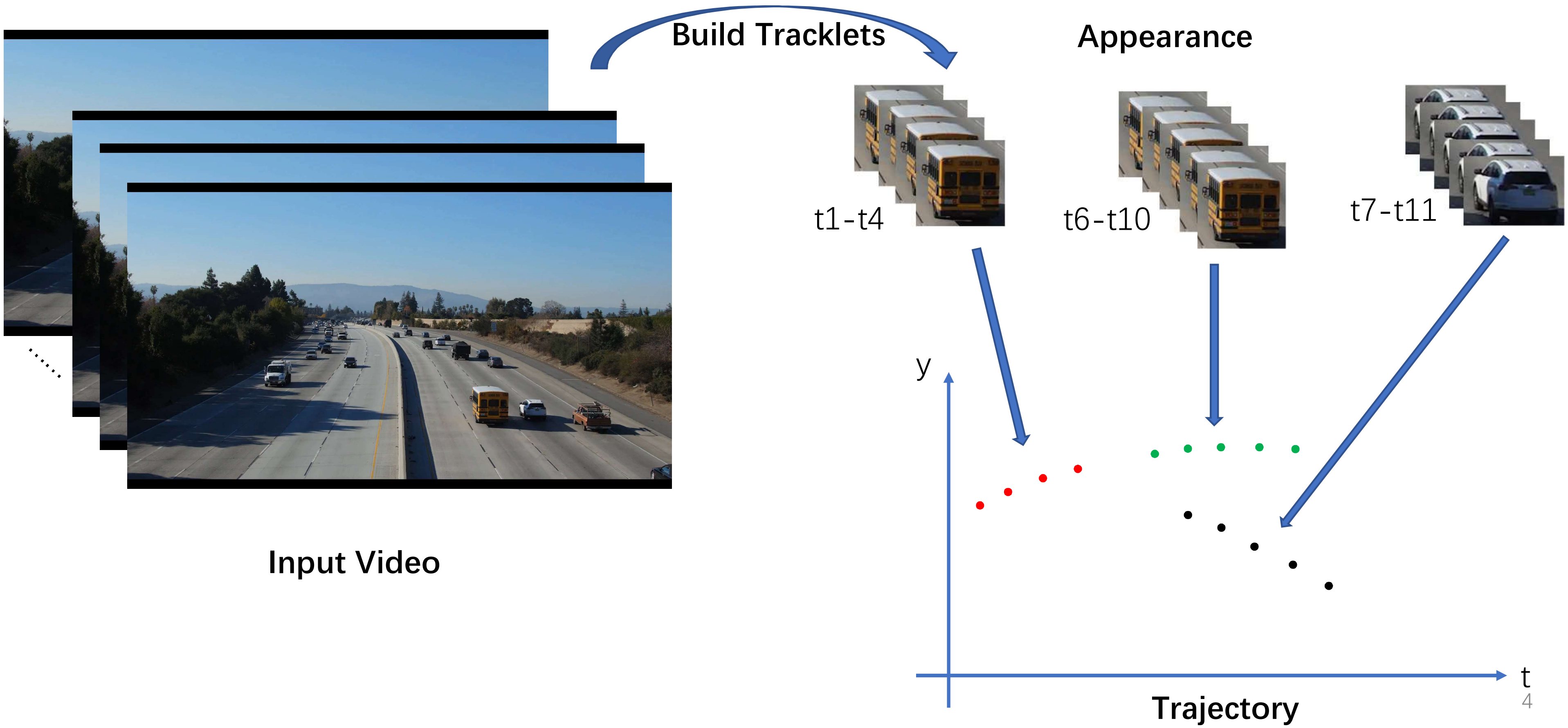
[1] Redmon, J., & Farhadi, A. (2017). YOLO9000: better, faster, stronger. *arXiv preprint*.

Challenges of Tracking by Detection



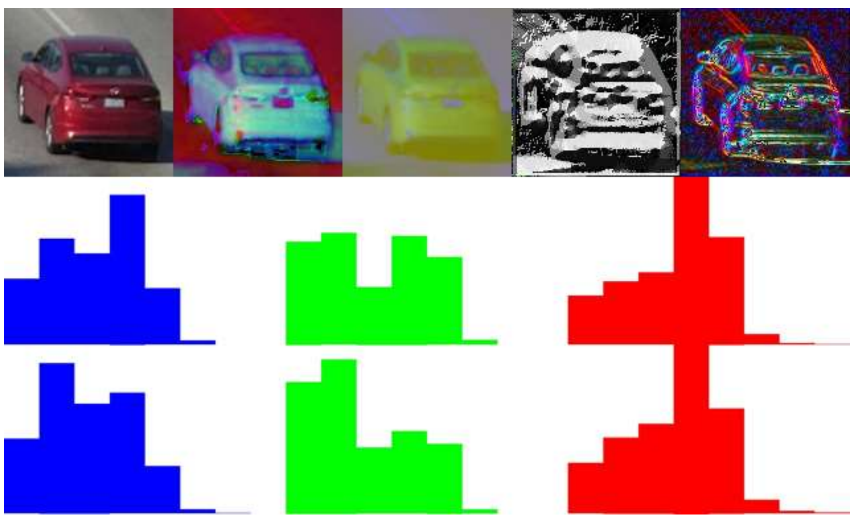
Challenges

Tracklet-based Clustering



Adaptive Appearance Modeling

- Histogram-based adaptive appearance model
  - A **history** of **spatially weighted (kernel)** histogram combinations will be kept for each vehicle



The first row respectively presents the RGB, HSV, Lab, LBP and gradient feature maps for an object instance in a tracklet, which are **used to build feature histograms**.
The second row shows the original RGB color histograms.
The third row demonstrates the Gaussian spatially weighted (kernel) histograms, where the contribution of background area is suppressed.

Tracklet-based Clustering

- Clustering Loss

Loss ?

$$l = \lambda_{\mathrm{sm}} l_{\mathrm{sm}} + \lambda_{\mathrm{ac}} l_{\mathrm{ac}} + \lambda_{\mathrm{ti}} l_{\mathrm{ti}}$$

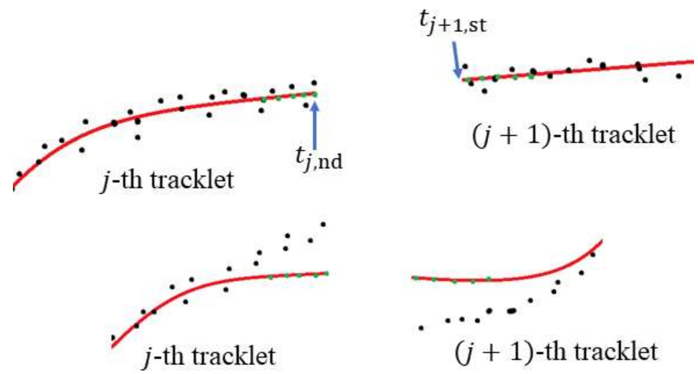
Smoothness in the trajectory

How far away in the time domain

Appearance change

Same trajectory

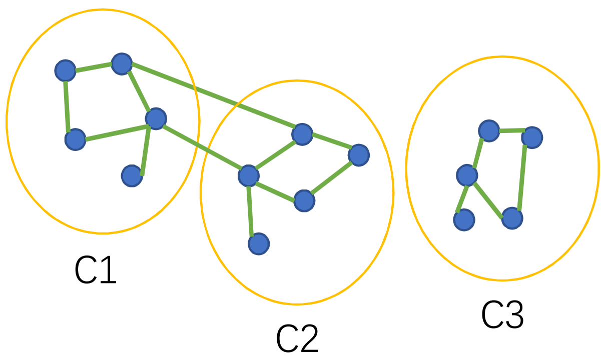
Different trajectory



Black dots show the detected locations at time *t*.
Red curves represent trajectories from Gaussian regression.
Green dots show n_{k} neighboring points on the red curves around the endpoints of the tracklets at $t_{j,\mathrm{nd}}$ and $t_{j+1,\mathrm{st}}$.

Tracklet-based Clustering

- **Edge** represents the clustering loss between two nodes (tracklets).



C: clusters.
Blue node: tracklet.
Green edge: clustering loss.

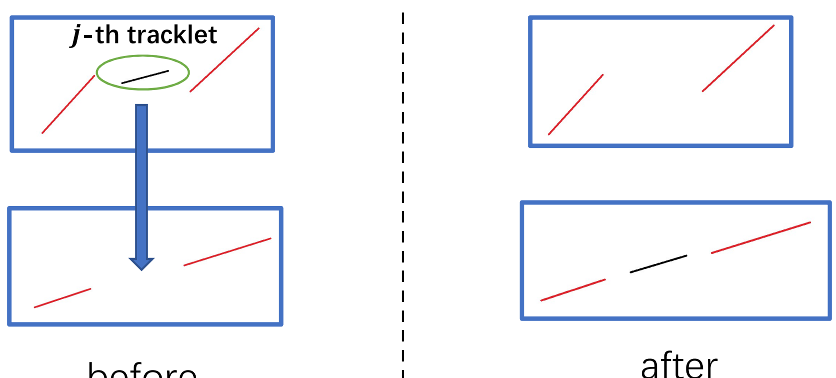
Optimization by Clustering

- A) Assign
  - Denote the trajectory set of the j-th tracklet as $S(j)$ which is a set of tracklets belonging to the trajectory. The loss change after assign operation can be expressed by,

$$\Delta l_a = \min_i \underbrace{\left(l(S(j)\backslash\{j\}) + l(S_i \cup \{j\}) \right)}_{\text{Loss after operation}} - \underbrace{\left(l(S(j)) + l(S_i) \right)}_{\text{Loss before operation}}$$

Cluster $S(j)$

Cluster $S(i)$



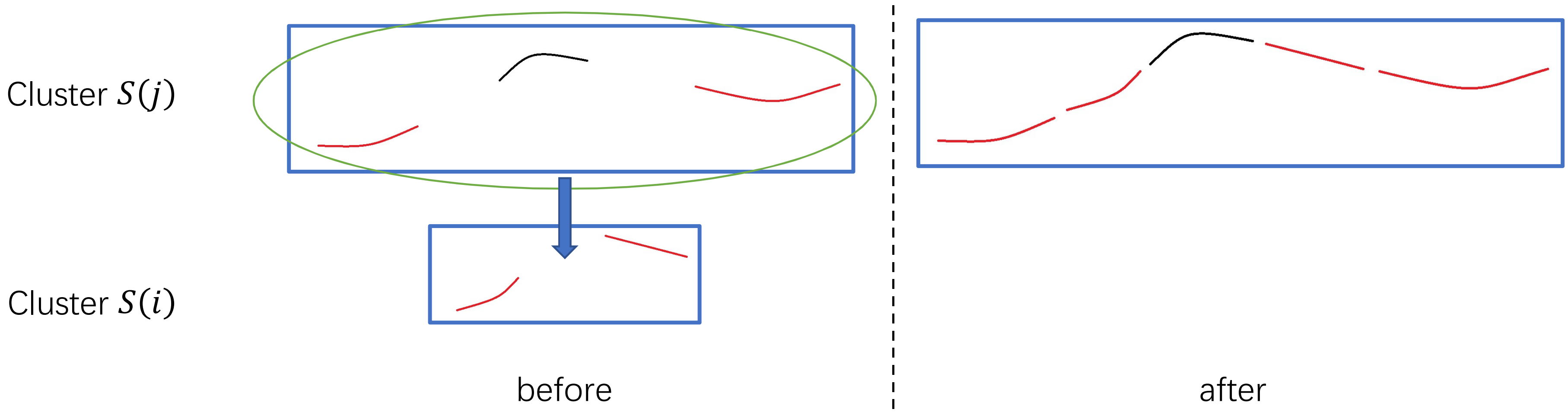
before

after

Optimization by Clustering

- B)   Merge

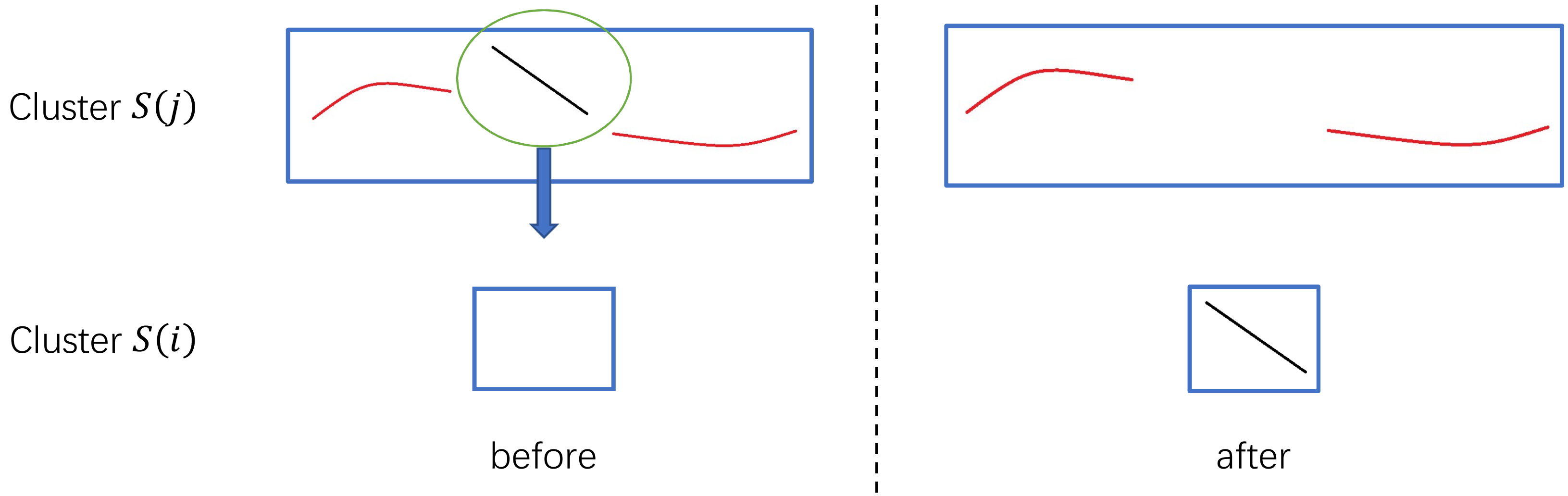
$$\Delta l_m = \min_i \left(l(S(j) \cup S_i) \right) - \left(l(S(j)) + l(S_i) \right)$$



Optimization by Clustering

- C) Split

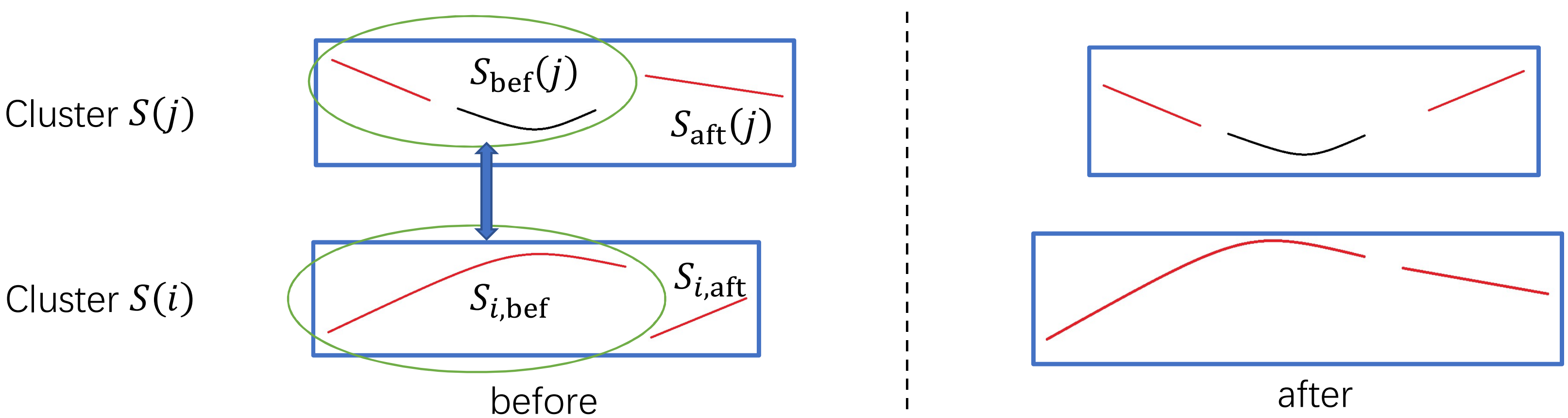
$$\Delta l_{sp} = \left(l(\{j\}) + l(S(j)\backslash\{j\})\right) - l\left(S(j)\right)$$



Optimization by Clustering

- D) Switch
  - For the j-th tracklet, denote all the tracklets in $S(j)$ after the j-th tracklet as $S_{\mathrm{aft}}(j)$ and other tracklets as $S_{\mathrm{bef}}(j)$. Then make the same splitting for all the trajectory set based on the j-th tracklet. Then we switch $S_{\mathrm{bef}}(j)$ and $S_{i,\mathrm{bef}}$ to calculate the loss change as follows,

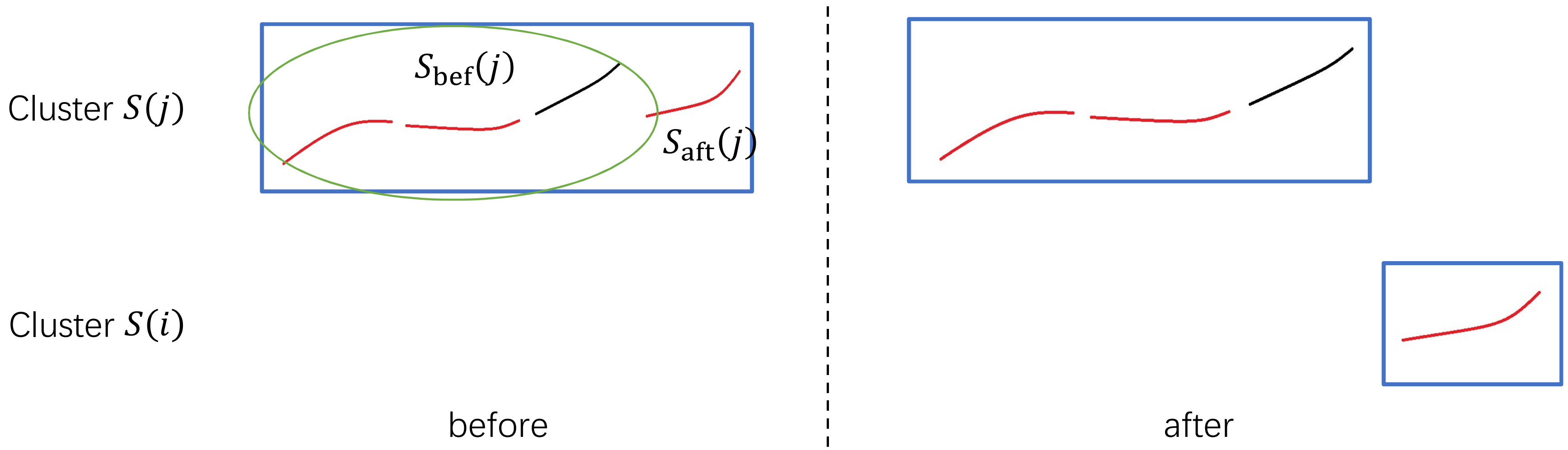
$$\Delta l_{sw} = \min_i \left(l\left(S_{bef}(j) \cup S_{i,aft}\right) + l\left(S_{aft}(j) \cup S_{i,bef}\right) \right) - \left(l\left(S(j)\right) + l(S_i) \right)$$



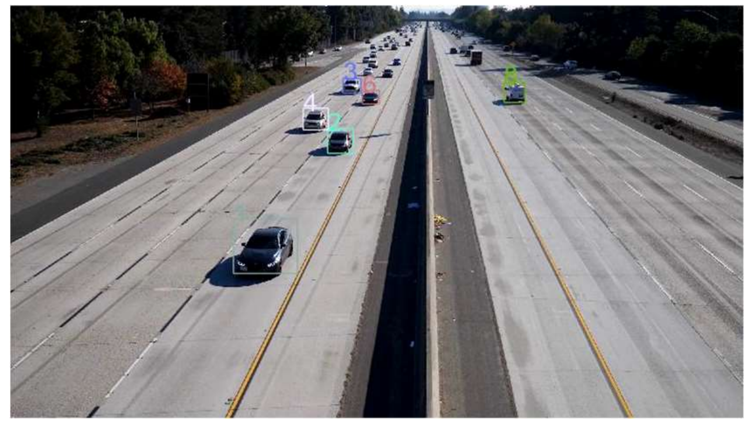
Optimization by Clustering

- E) Break

$$\Delta l_b = \left(l\left(S_{bef}(j)\right) + l\left(S_{aft}(j)\right)\right) - l(S(j))$$



Resulting Trajectories from Tracklets



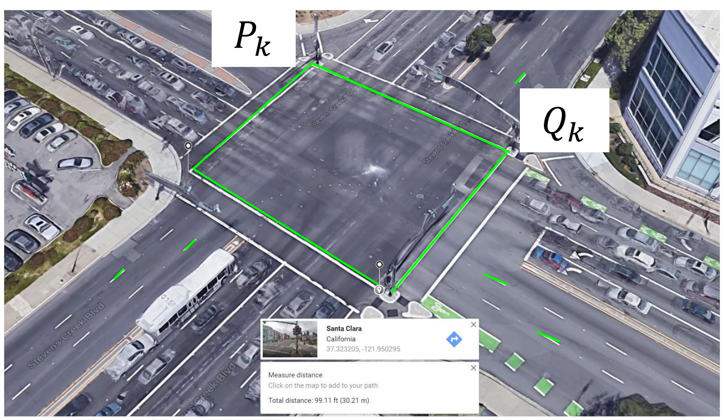
Camera Calibration

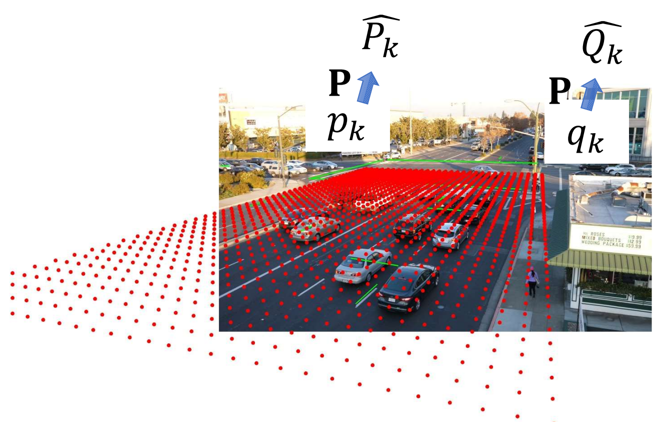
- Minimization of reprojection error solved by EDA

$$\min_{\mathbf{P}} \sum_{k=1}^{N_{\mathrm{ls}}} \left| \|P_k - Q_k\|_2 - \left\|\widehat{P_k} - \widehat{Q_k}\right\|_2 \right|$$

$$\mathrm{s.t.}\, \mathbf{P} \in \mathrm{Rng}_{\mathbf{P}}, p_k = \mathbf{P} \cdot \widehat{P_k}, q_k = \mathbf{P} \cdot \widehat{Q_k}$$

$\mathbf{P}$: Camera projection matrix
$\mathrm{Rng}_{\mathbf{P}}$: Range for optimization
P_k, Q_k: True endpoints of line segments
$\widehat{P_k}$, $\widehat{Q_k}$: Estimated endpoints of line segments
p_k, q_k: 2D endpoints of line segments
N_{ls}: Number of endpoints





Results on AI City Challenge 2018 (Track 1) [1,2]

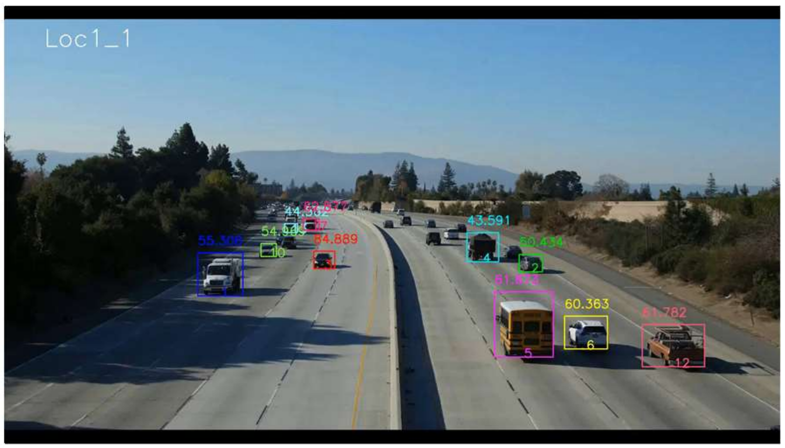
- Track 1 - Traffic flow analysis
  - 27 videos, each 1 minute in length, recorded at 30 fps and 1080p resolution
  - Performance evaluation: $S1 = DR \times (1 - NRMSE)$
  - *DR* is the detection rate and *NRMSE* is the normalized Root Mean Square Error (RMSE) of speed
- 56 teams participated, 13 teams submitted the final results.

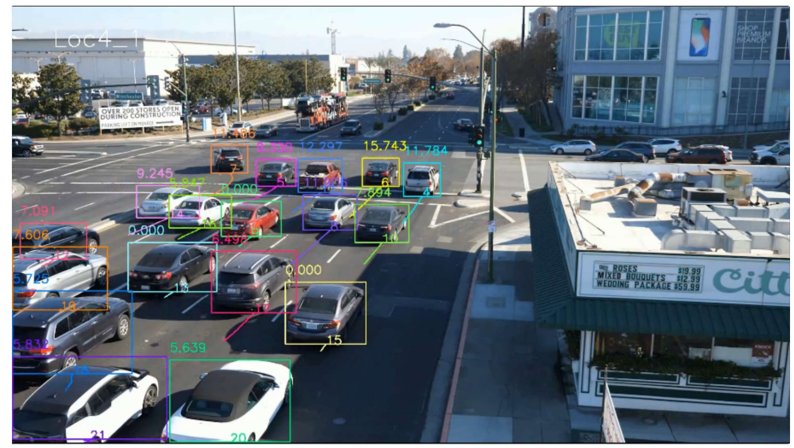
[1] Naphade, M., Chang, M. C., Sharma, A., Anastasiu, D. C., Jagarlamudi, V., Chakraborty, P., ... & Hwang, J. N. (2018). The 2018 NVIDIA AI City Challenge. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 53-60).
[2] Tang, Z., Wang, G., Xiao, H., Zheng, A., & Hwang, J. N. (2018). Single-camera and inter-camera vehicle tracking and 3D speed estimation based on fusion of visual and semantic features. In *CVPR Workshop (CVPRW) on the AI City Challenge*.

Results on AI City Challenge 2018 (Track 1)

Table 1. Quantitative comparison of speed estimation on the *NVIDIA AI City Dataset* [9]

Rank	Team	S1 Score
1	team48	1.0000
2	team79	0.9162
3	team78	0.8892
4	team24	0.8813
5	team12	0.8331
6	team4	0.7924
7	team65	0.7654
8	team6	0.7174
9	team40	0.6564
10	team26	0.6547
11	team18	0.6226
12	team45	0.5953
13	team39	0.0000

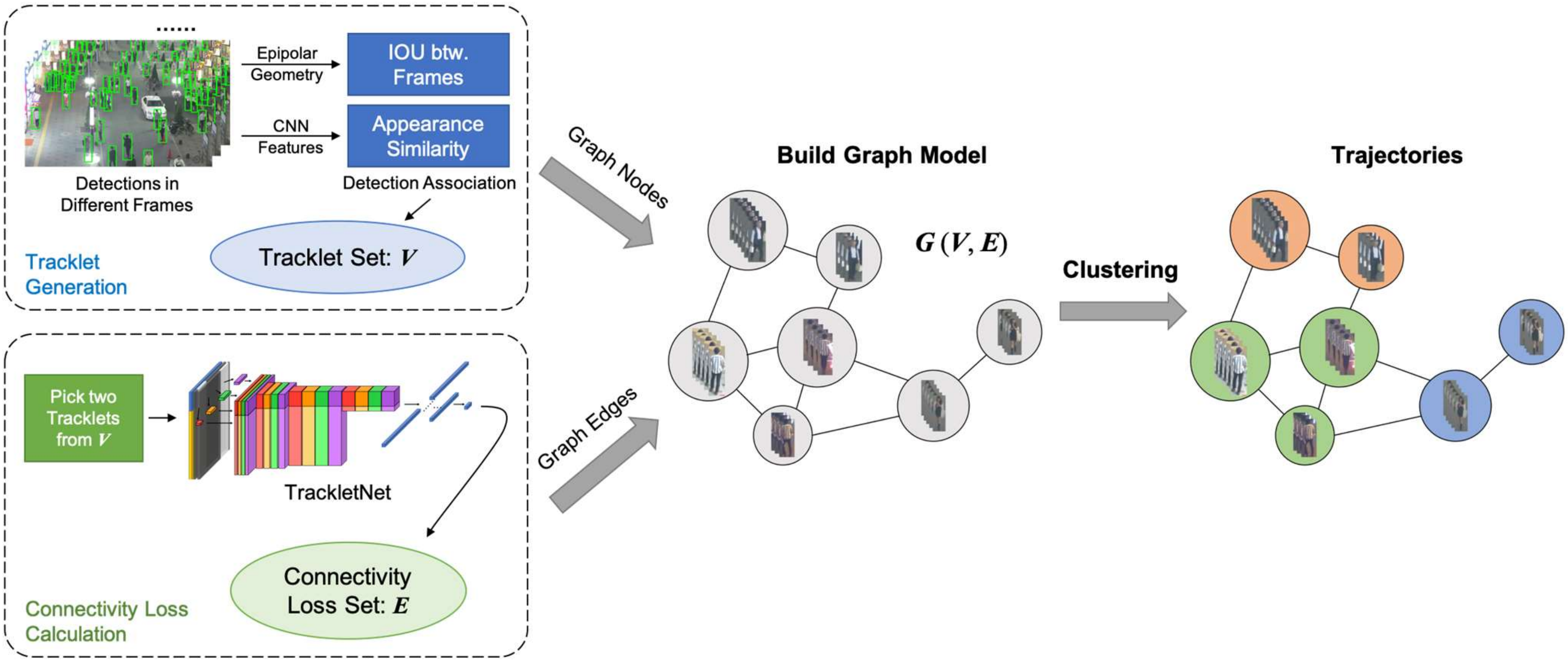




DR: 1.0000 RMSE: 4.0963 mi/h

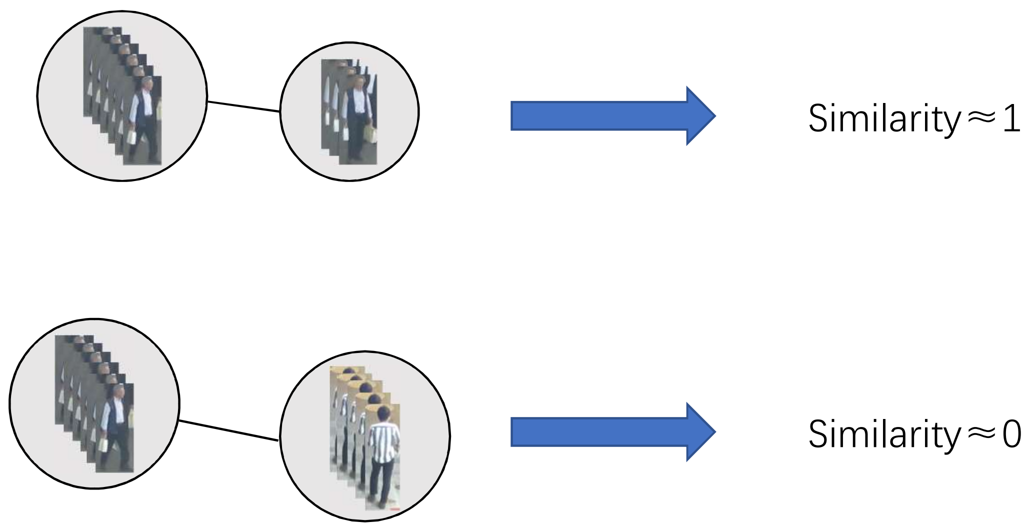
Acknowledgement
We thank NVIDIA for organizing AI City Challenge and providing the dataset for training and evaluation.

General Multi-Object Tracking (Ongoing)

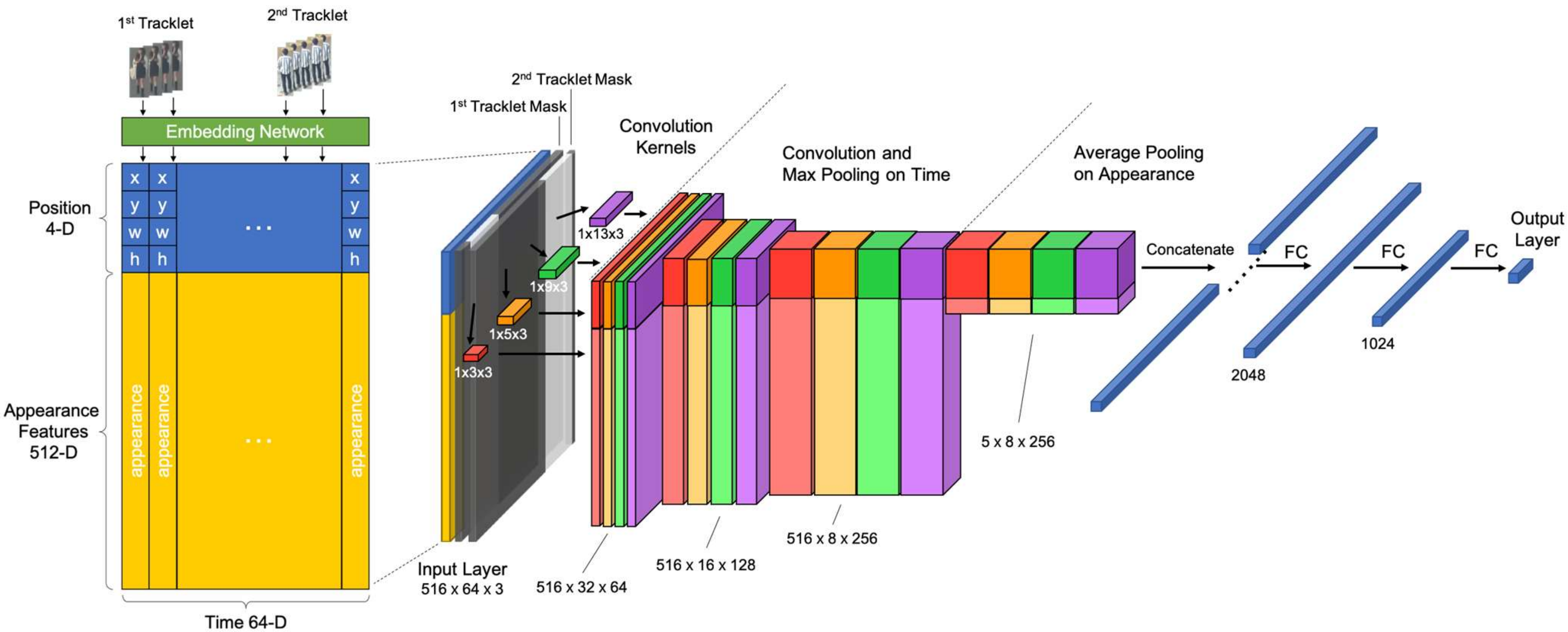


Connectivity Loss

- The loss for merging two tracklets
  - Same ID
  - Different ID



TrackletNet



TrackletNet

- Input tensor (B x D x T x C)
  - B (32): batch size in the training.
  - D (516): feature dimension for each detection.
    - 4-D Location feature: x, y, width, height.
    - 512-D appearance feature: learned from FaceNet [1].
  - T (64): time window
  - C (3): input channels
    - C_1: two embedded tracklet feature map.
    - C_2: binary mask to indicate the location of 1^{st} tracklet.
    - C_3: binary mask to indicate the location of 2^{nd} tracklet.

[1] Schroff, Florian, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

TrackletNet

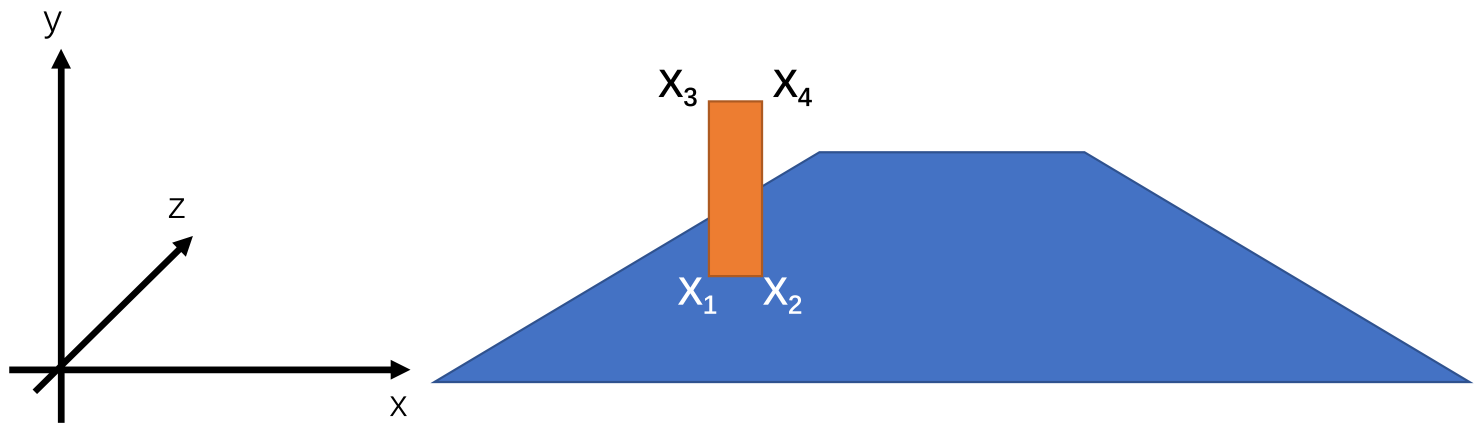
- Architecture
  - **4 sizes of convolution kernels**: 1 x 3, 1 x 5, 1 x 9, 1 x 13. Different kernels can deal with different lengths of missing detections.
  - **3 convolution and max pooling layers**: feature extraction.
  - **1 average pooling** on appearance dimensions after the last max pooling: weighted majority vote on 512 dimensions of appearance features to measure the appearance change.
  - **2 fully connected layers**.

Properties of TrackletNet

- Convolution along time domain only.
  - No convolution across feature space.
  - The complexity of the network is largely reduced, which can address overfitting.
- Convolution solves connectivity loss.
  - Convolution includes lowpass and highpass filters in time domain.
  - Lowpass filters can suppress the detection noise.
  - Highpass filters can measure whether there are abrupt changes.
- Binary masks can tell the missing detections to the network.

Convert to 3D Tracking

- $(x_{2d}, y_{2d}, w_{2d}, h_{2d}) \rightarrow (x_{3d}, y_{3d}, z_{3d}, w_{3d}, h_{3d})$
- Obtain 3D
  - Estimate foot location $(x_{3d}, y_{3d}, z_{3d}) = (X_2+X_1)/2$ by ground plane.
  - $w_{3d}=||X_2-X_1||$, $h_{3d}= w_{3d} \times h_{2d}/ w_{2d}$
  - Detection location $(x_{3d}, z_{3d}, w_{3d}, h_{3d})$. Drop y_{3d} out since (x_{3d}, y_{3d}, z_{3d}) are linear dependent.



TrackletNet Training (2D and 3D)

- The same input size of 2D and 3D tracking. They share the same architecture.
- Augmentation in training
    - Bounding box randomization
        - Randomly disturb the size and location of bounding boxes by a factor of random noise sampled from normal distribution with mean and standard deviation to be 0 and 0.05, respectively.
    - Tracklet random split
        - Randomly divide each trajectory into small pieces of tracklets.
    - Tracklet random combination
        - Randomly select two tracklets as the input of the network.

MOT Challenge 2016[1]

[1] Milan, A., Leal-Taixé, L., Reid, I., Roth, S., & Schindler, K. (2016). MOT16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*.

Results on MOT Benchmark

MOT16

Tracker	IDF1	MOTA	MT	ML	FP	FN	ID sw	Frag
Ours	56.1	49.2	17.3%	40.3%	8,400	83,702	606	882
TLMHT	55.3	48.7	15.7%	44.5%	6,632	86,504	413	642
DMMOT	54.8	46.1	17.4%	42.7%	7,909	89,874	532	1,616
NOMT	53.3	46.4	18.3%	41.4%	9,753	87,565	359	504
eHAF16	52.4	47.2	18.6%	42.8%	12,586	83,107	542	787

MOT17

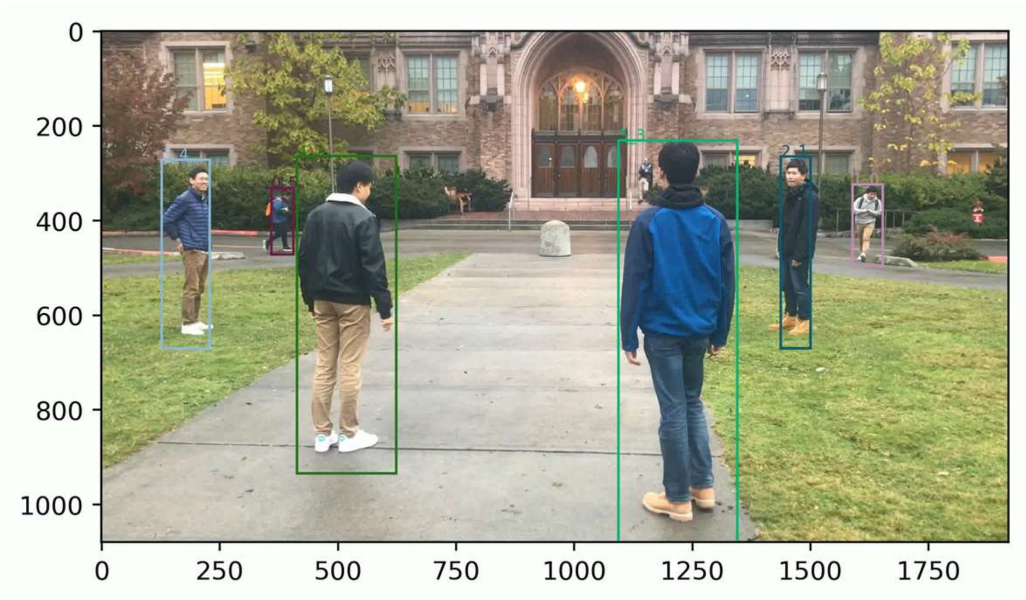
Tracker	IDF1	MOTA	MT	ML	FP	FN	ID sw	Frag
Ours	58.0	51.9	23.5%	35.5%	37,311	231,658	2,294	2,917
TLMHT	56.5	50.6	17.6%	43.4%	22,213	255,030	1,407	2,079
DMAN	55.7	48.2	19.3%	38.3%	26,218	263,608	2,194	5,378
eHAF17	54.7	51.8	23.4%	37.9%	33,212	236,772	1,834	2,739
jCC	54.5	51.2	20.9%	37.0%	25.937	247,822	1,802	2,984

Examples and Applications

- 1. 3D Pose estimation
  - Not many works related to multi-person pose estimation.
  - Not many works dealing with missing pose and occlusions.
  - Tracking can be treated as a preprocessing step to the above issues.

Examples and Applications

- Use pre-trained model on MOT without finetune.
- Detection: OpenPose

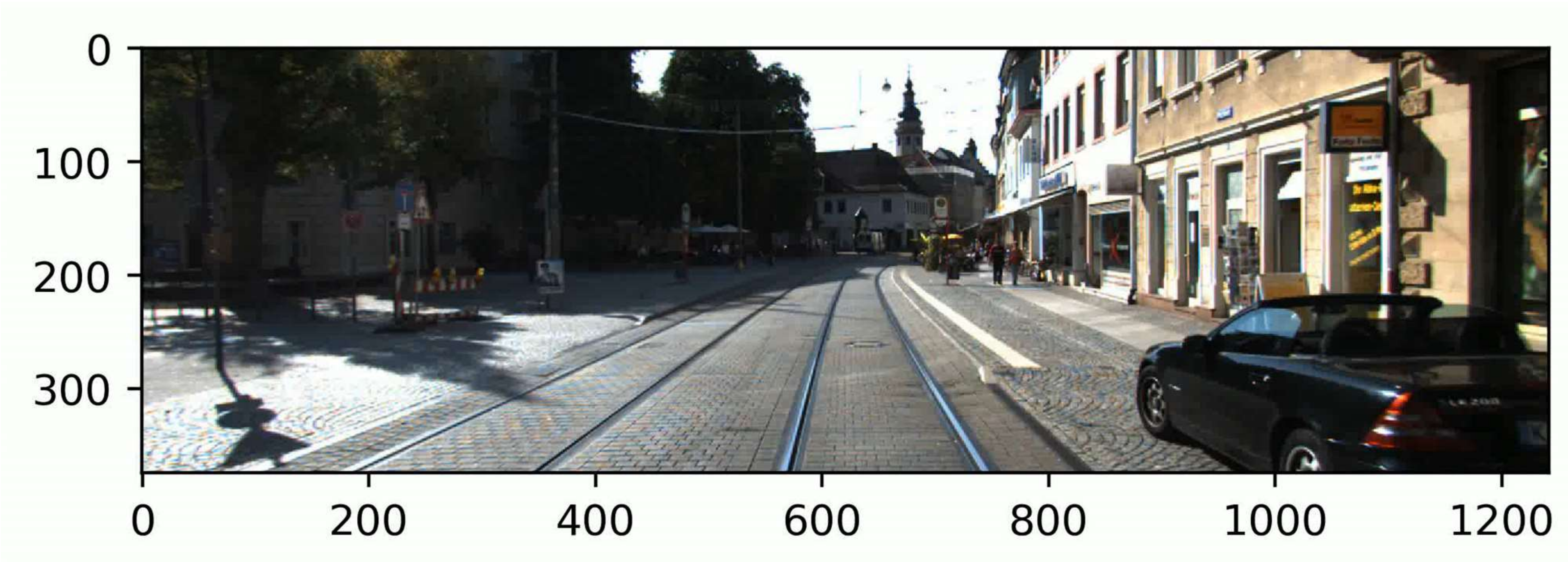


Examples and Applications

- 2. Autonomous Driving
  - Estimate the speed of pedestrian.
  - Anomaly detection of person behaviors.
  - Tracking can be also involved in ground plane estimation and bundle adjustment as additional constraints.

Examples and Applications

- Detection: Yolo



Examples and Applications

- 3. Drone applications
  - Similar to autonomous driving.
- Use pre-trained model on KITTI without finetune.
- Detection: Mask-RCNN