CS 498: Assignment 5: 3D Multi Object Tracking

mukaiyu2

April 17, 2022

Submission

You will be submitting two files, "kalman_filter.py" and "matching.py". Please put together a single PDF with your answers and figures for each problem, and submit it to Gradescope (Course Code: JBXJVZ). We recommend you add your answers to the latex template files we provided. More details on what to report are in the provided code.

Reminder: please put your name and netid in your pdf.

Provided Files

Files you will modify:

- kalman_filter.py:
  - define_model (define dynamics)
  - update (observation model)
  - predict (state propagation)
- matching.py (greedy matching algorithm)
- main.py (for visualization and debugging)

Files you will not (need to) modify:

- evaluate.py: run this after running main.py to do evaluation, i.e. "python evaluate.py"
- kitti_calib/oxts.py: these are used for something called ego-motion-compensation, explained in code
- matching_utils.py: contains the function iou(box_a, box_b) which you will need to compute similarity when doing matching
- utils.py: some utils used by main.py, you should look at Box3D
- vis.py: some code for visualizing the data. You only really need vis_obj, which is described more clearly in main.py

File structure

```
data
├── calib
│   └── training
│       └── [seq_name].txt
├── detection
│   └── [seq_name].txt
├── label
│   └── [seq_name].txt
├── oxts
│   └── training
│       └── [seq_name].txt
├── image_02
│   └── training
│       └── [seq_name]
│           └── [frame_num].png
└── results
    ├── eval (files in here will be created automatically)
    └── img_vis (files in here will be created automatically)
```

Multi Object Tracking (MOT)

In this assignment, you will implement a multi object tracker for 3D objects. In other words, given a sequence of frames taken from the perspective of a car, track the other cars in the images. In this project we will develop our tracking algorithm on KITTI dataset(`http://www.cvlibs.net/datasets/kitti/`).

The idea is as follows: we can use a good pre-trained object detector to find objects in each frame (we've already done that for you, check `https://github.com/sshaoshuai/PointRCNN` if you want to know more). Now, simply identifying objects is not good enough, so we want to track them across frames for consistency. You will implement two main methods which together will do this quite well.

The first is a matching algorithm: given a list of 3D bounding boxes for objects detected by the object detector in the current frame, match them to objects you are already tracking from the previous frame.

The second is a Kalman Filter. Just matching current detections to past trackers is not great since cars can move and therefore the previous and current bounding boxes will not overlap perfectly (this is especially problematic if an object becomes occluded for a few frames). To deal with this issue you will implement a Kalman Filter for forward propagating each object. Moreover, you will now use each object detection to "update" the state of your tracked object, as an observation to the filter.

Question 0 (Inspect the Data)[1 pt]: Before you begin the actual assignment, read the code we've provided. Namely read through "main.py" so you understand the basic structure of the algorithm and the functionality of each component. You will need to modify main.py, but only for debugging/visualization purposes.

For this question, please visualize the detections for frame 100 of the first and second sequences, i.e. sequence 0000 and sequence 0001. Each detection should be a different color. You should work off the code we provided in the Visualization section of main.py.

Please download the images for the first sequence (0000), and put them in the right place in the directory structure above. Use your illinois address to access the drive link: `https://drive.google.com/file/d/15lWATvV4p9UCShnnPa2SEL8BTh2twIm4/view?usp=sharing`

Question 1 (Greedy Matching)[3 pt]: For this question you will be modifying "matching.py". You should implement a greedy matching approach, whereby you match pairs of detections and tracked objects in order of similarity. First match the detection and tracker that are most similar, then remove them from the set, and continue, until you have no trackers or no detections. Also, if the similarity for a match is more than the provided threshold, do not consider the pair a match.

Notice we provide you with "iou(box_a, box_b)" to compute similarity between 3D detected regions.

Question 2 (Trivial Update) [1 pt]: You'll notice that even though you've implemented matching, the trackers themselves don't update location. For this question you will implement a trivial update to each tracker in kalman_filter.py. Given a matched detection z, simply set the associated tracker to have the same bounding box z. In other words, we simply trust the observation 100% when updating the state, neglecting any motion or noise models.

In your pdf please show your code for this part, it should be very simple. Report your evaluation MOTA for this setting (meaning matching=greedy, predict() is unimplemented, and the update is trivial).

Question 2 Answer:

```
# ------------------------- Begin your code here ---------------------------------------------
matches = []
unmatched_dets = np.ones(len(dets))
unmatched_trks = np.ones(len(trks))

gious = np.zeros((len(dets), len(trks)))
for i in range(len(dets)):
    for j in range(len(trks)):
        gious[i, j] = iou(dets[i], trks[j])

while len(dets) > 0 and len(trks) > 0 and np.max(gious) >= threshold:
    idx = np.unravel_index(np.argmax(gious), gious.shape)
    unmatched_dets[idx[0]] = 0  # mask as selected
    unmatched_trks[idx[1]] = 0  # mask as selected
    matches.append([idx[0], idx[1]])

    gious[idx[0], :] = - 2.0  # mask all selected matches
    gious[:, idx[1]] = - 2.0

unmatched_dets = np.array(*np.where(unmatched_dets > 0)).astype(int)
unmatched_trks = np.array(*np.where(unmatched_trks > 0)).astype(int)

matches = np.array(matches)
# ------------------------- End your code here   ---------------------------------------------

return matches, unmatched_dets, unmatched_trks
```

data_association()

```python
def update(self, z):
    """
    Add a new measurement (z) to the Kalman filter.
    ----------
    z : (dim_z, 1): array_like measurement for this update.
    """
    z = z.reshape(-1,1)

    # ---------------------------- Begin your code here -----------------------------------------------
    self.x[:7] = z.copy()

    # ---------------------------- End your code here   -----------------------------------------------

    # Leave this at the end, within_range ensures that the angle is between -pi and pi
    self.x[3] = within_range(self.x[3])
    return
```

update()

```
================evaluation: best results with single threshold================
Multiple Object Tracking Accuracy (MOTA)                              0.8419
Multiple Object Tracking Precision (MOTP)                             0.8302
Multiple Object Tracking Accuracy (MOTAL)                             0.8419
Multiple Object Detection Accuracy (MODA)                             0.8419
Multiple Object Detection Precision (MODP)                            0.9525

Recall                                                                0.9929
Precision                                                             0.9310
F1                                                                    0.9609
False Alarm Rate                                                      0.2013

Mostly Tracked                                                        1.0000
Partly Tracked                                                        0.0000
Mostly Lost                                                           0.0000

True Positives                                                         418
Ignored True Positives                                                 206
False Positives                                                         31
False Negatives                                                          3
Ignored False Negatives                                                114
ID-switches                                                              0
Fragmentations                                                           1

Ground Truth Objects (Total)                                           535
Ignored Ground Truth Objects                                           320
Ground Truth Trajectories                                               12

Tracker Objects (Total)                                                524
Ignored Tracker Objects                                                 75
Tracker Trajectories                                                    44
================================================================================
======================evaluation: average over recall=======================
 sAMOTA  AMOTA  AMOTP
0.9552 0.5610 0.8414
================================================================================
```

Result is pretty good even without Kalman Filter, thanks to pointRCNN

Question 3 (Kalman Linear Dynamics) [3 pt]: For this part you will fill in define_model() in the class Kalman. The state **x** should consist three dimensional box center, raw angle, three dimensional box size, and finally three dimensional linear velocity (total 10D). The motion model is a constant linear velocity model in 3D. Your model should be linear, meaning x' = x + dx = Ax. In addition you should define the measurement model and measurement uncertainty, meaning H and Sigma and Q. In your pdf please report A, H, Sigma, Q, and R. Explain why each is set the way it is.

Question 3 Answer:

$$\text{A} = \begin{bmatrix} 1 & & & & & & & 1 & & \\ & 1 & & & & & & & 1 & \\ & & 1 & & & & & & & 1 \\ & & & 1 & & & & & & \\ & & & & 1 & & & & & \\ & & & & & 1 & & & & \\ & & & & & & 1 & & & \\ & & & & & & & 1 & & \\ & & & & & & & & 1 & \\ & & & & & & & & & 1 \end{bmatrix} \in \mathbb{R}^{dim\_x \times dim\_x}, \text{ because we have velocity model } \begin{cases} x' = x + dx \\ y' = y + dy \\ z' = z + dz \end{cases},$$

while all others (theta, l, w, h, dx, dy, dz) remain the same.

$$\text{H} = \begin{bmatrix} 1 & & & & & & & & & \\ & 1 & & & & & & & & \\ & & 1 & & & & & & & \\ & & & 1 & & & & & & \\ & & & & 1 & & & & & \\ & & & & & 1 & & & & \\ & & & & & & 1 & & & \\ & & & & & & & 1 & & \\ & & & & & & & & 1 & \\ & & & & & & & & & 1 \end{bmatrix} \in \mathbb{R}^{dim\_z \times dim\_x}, \text{ because standard linear measurement is used.}$$

$$\Sigma = \begin{bmatrix} 1 & & & & & & & & & \\ & 1 & & & & & & & & \\ & & 1 & & & & & & & \\ & & & 1 & & & & & & \\ & & & & 1 & & & & & \\ & & & & & 1 & & & & \\ & & & & & & 1 & & & \\ & & & & & & & 1 & & \\ & & & & & & & & 5 & \\ & & & & & & & & & 5 \end{bmatrix} \in \mathbb{R}^{dim\_x \times dim\_x}, \text{ because initial state uncertainty is huge, even larger}$$

for velocity state.

$$\text{Q} = \begin{bmatrix} 0.3 & & & & & & & & & \\ & 0.3 & & & & & & & & \\ & & 0.3 & & & & & & & \\ & & & 0.3 & & & & & & \\ & & & & 0.3 & & & & & \\ & & & & & 0.3 & & & & \\ & & & & & & 0.3 & & & \\ & & & & & & & 0.3 & & \\ & & & & & & & & 0.3 & \\ & & & & & & & & & 0.3 \end{bmatrix} \in \mathbb{R}^{dim\_x \times dim\_x}, \text{ because prediction uncertainty}$$

should be less than initial state uncertainty.

$$R = \begin{bmatrix} 0.1 & & & & & & \\ & 0.1 & & & & & \\ & & 0.1 & & & & \\ & & & 0.1 & & & \\ & & & & 0.1 & & \\ & & & & & 0.1 & \\ & & & & & & 0.1 \end{bmatrix} \in \mathbb{R}^{dim\_z \times dim\_z}$$, because pointRCNN output is the only sensor measurement signal we have, and we choose to trust it more than prediction.

Question 4 (Kalman Update) [3 pt]: Now implement a proper Kalman Filter Update step, where you use a matched object detection as a noisy observation for updating the state. See lecture 10-12 for more details.

In your pdf please describe the Kalman Filter linear update mathematically and report your evaluation MOTA under this setting (matching=greedy, predict() unimplemented, update implemented).

Question 4 Answer:

```python
def update(self, z):
    """
    Add a new measurement (z) to the Kalman filter.
    ----------
    z : (dim_z, 1): array_like measurement for this update.
    """
    z = z.reshape(-1,1)

    # --------------------------- Begin your code here ---------------------------------------------
    # self.x[:7] = z.copy() # Trivial Update

    self.z = z.copy()

    self.y = self.H @ self.x # y used as \miu_{t + 1}
    self.S = self.H @ self.Sigma @ self.H.T + self.R
    self.SI = np.linalg.inv(self.S)

    self.K = self.Sigma @ self.H.T @ self.SI
    self.x = self.x + self.K @ (self.z - self.y)
    self.Sigma = self.Sigma - self.K @ self.H @ self.Sigma

    print("ID", self.ID)
    print("miu =", self.x)
    print("Sigma =", self.Sigma)
    # --------------------------- End your code here   ---------------------------------------------

    # Leave this at the end, within_range ensures that the angle is between -pi and pi
    self.x[3] = within_range(self.x[3])
    return
```

update() implementation

update():

$$\mu_z \leftarrow H \times x$$
$$S \leftarrow H \times \Sigma \times H^T + R$$
$$K \leftarrow \Sigma \times H^T \times S^{-1}$$

$$x \leftarrow x + K \times (z - \mu_z)$$
$$\Sigma \leftarrow \Sigma - K \times H \times \Sigma$$

MOTA result:

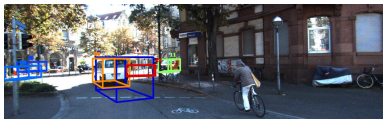
```
==================evaluation: best results with single threshold================
Multiple Object Tracking Accuracy (MOTA)                              0.0558
Multiple Object Tracking Precision (MOTP)                             0.7899
Multiple Object Tracking Accuracy (MOTAL)                             0.0558
Multiple Object Detection Accuracy (MODA)                             0.0558
Multiple Object Detection Precision (MODP)                            0.9563

Recall                                                                0.1488
Precision                                                             0.6154
F1                                                                    0.2397
False Alarm Rate                                                      0.1299

Mostly Tracked                                                        0.0000
Partly Tracked                                                        0.2222
Mostly Lost                                                           0.7778

True Positives                                                          32
Ignored True Positives                                                   0
False Positives                                                         20
False Negatives                                                        183
Ignored False Negatives                                                320
ID-switches                                                              0
Fragmentations                                                           0

Ground Truth Objects (Total)                                           535
Ignored Ground Truth Objects                                           320
Ground Truth Trajectories                                               12

Tracker Objects (Total)                                                 55
Ignored Tracker Objects                                                  3
Tracker Trajectories                                                    58
================================================================================
========================evaluation: average over recall=========================
 sAMOTA  AMOTA  AMOTP
0.0326 -0.2798 0.4460
================================================================================
```

result is very bad

Reason Analysis:

initial tracking at frame 0

lagged tracking at frame 20



new tracking at frame 22

We can see from the tracking of the van that the tracker cannot keep up with the detection even when the measurement uncertainty is small, it's hard for the measurement to drag the overall state.
Printing out x and Σ we can see that Σ is decreasing with time, but the velocity in x is always 0 since no prediction is conducted, and center of the box only moves a little bit between frames, until it's too far from the new detection and cannot be matched.

Question 5 (Kalman Predict) [2 pt]: Up until now, each frame the detections were compared to each tracker, and then matched trackers were updated. But our matching is poor because the detections and trackers do not overlap (they are one frame apart). In this question you will implement the Kalman Filter Predict step, where you forward propagate the state according to the dynamics model you defined earlier.

In your pdf please describe the predict step, and report your evaluation MOTA under this setting (matching=greedy, predict and update both implemented).

Question 5 Answer:

```
def predict(self):
    """
    Predict next state (prior) using the Kalman filter state propagation
    equations.
    Parameters
    ----------
    Note that often we might give predict a control vector u,
    but of course we don't know what controls each tracked object is applying
    """
    # Hint: you should be modifying self.x and self.Sigma
    # -------------------------- Begin your code here ---------------------------------------------
    self.x = self.A @ self.x

    self.Sigma = self.A @ self.Sigma @ self.A.T + self.Q
    # -------------------------- End your code here   ---------------------------------------------

    # Leave this at the end, within_range ensures that the angle is between -pi and pi
    self.x[3] = within_range(self.x[3])
    return
```

predict() implementation

predict():

$$x \leftarrow A \times x$$
$$\Sigma \leftarrow A \times \Sigma \times A^T + Q$$

MOTA result:

```
=================evaluation: best results with single threshold================
Multiple Object Tracking Accuracy (MOTA)                               0.8465
Multiple Object Tracking Precision (MOTP)                              0.8345
Multiple Object Tracking Accuracy (MOTAL)                              0.8465
Multiple Object Detection Accuracy (MODA)                              0.8465
Multiple Object Detection Precision (MODP)                             0.9541

Recall                                                                 0.9906
Precision                                                              0.9354
F1                                                                     0.9622
False Alarm Rate                                                       0.1883

Mostly Tracked                                                         1.0000
Partly Tracked                                                         0.0000
Mostly Lost                                                            0.0000

True Positives                                                          420
Ignored True Positives                                                  209
False Positives                                                          29
False Negatives                                                           4
Ignored False Negatives                                                 111
ID-switches                                                               0
Fragmentations                                                            2

Ground Truth Objects (Total)                                            535
Ignored Ground Truth Objects                                            320
Ground Truth Trajectories                                                12

Tracker Objects (Total)                                                 524
Ignored Tracker Objects                                                  75
Tracker Trajectories                                                     44
================================================================================
======================evaluation: average over recall=========================
 sAMOTA  AMOTA  AMOTP
0.9578 0.5677 0.8461
================================================================================
```

result is very good compared to no prediction, Q/R = 1

Trust prediction and measurement equally makes the best result.

initial tracking at frame 0

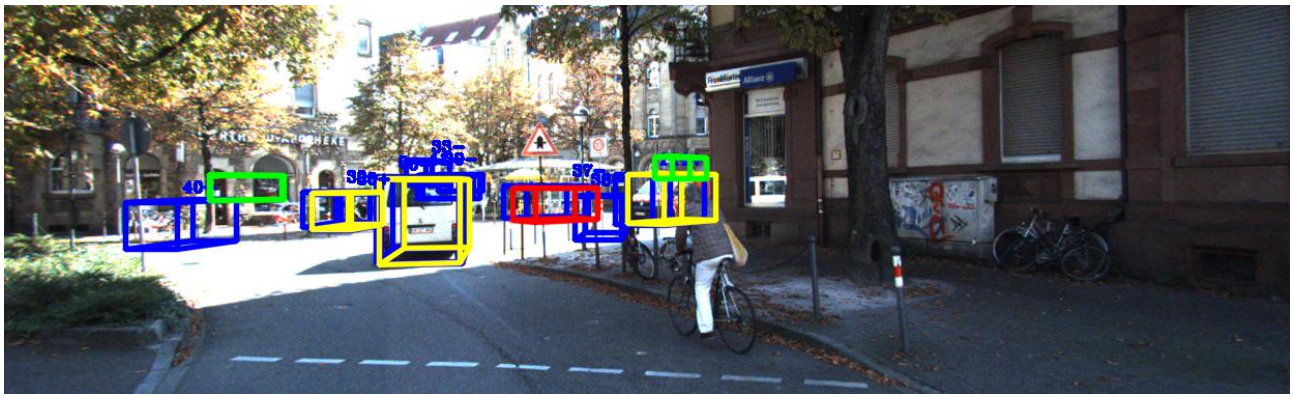
tracking at frame 20

tracking at frame 22, still keeps up

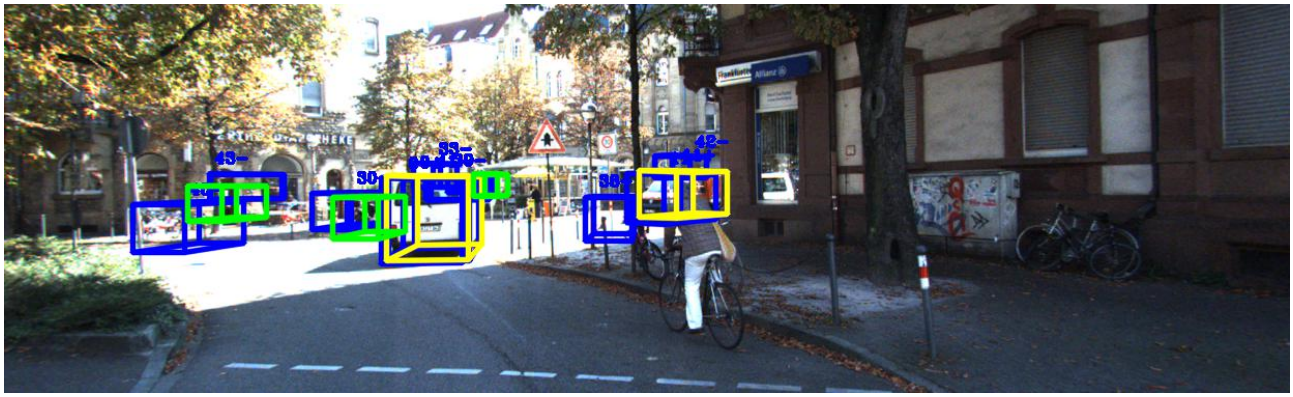
Question 6 (Final Visualization) [1 pt]: Please visualize some results from your final code. Pick at least 4 consecutive frames from sequence 0000. For each frame visualize all in one image:

- Show birthed trackers in green
- Show dead trackers in red
- For the rest of the trackers (these were matched), show each before predict and after update. Show their corresponding detections. Color the trackers in blue and detections in yellow. Add text above each tracker with its ID and the text "-" for before predict or "+" for after update.

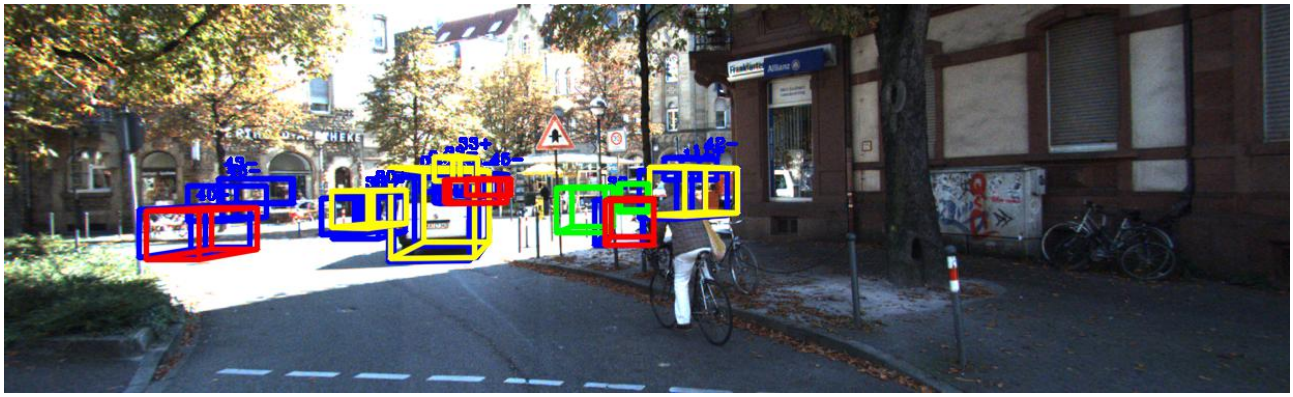
Question 6 Answer:



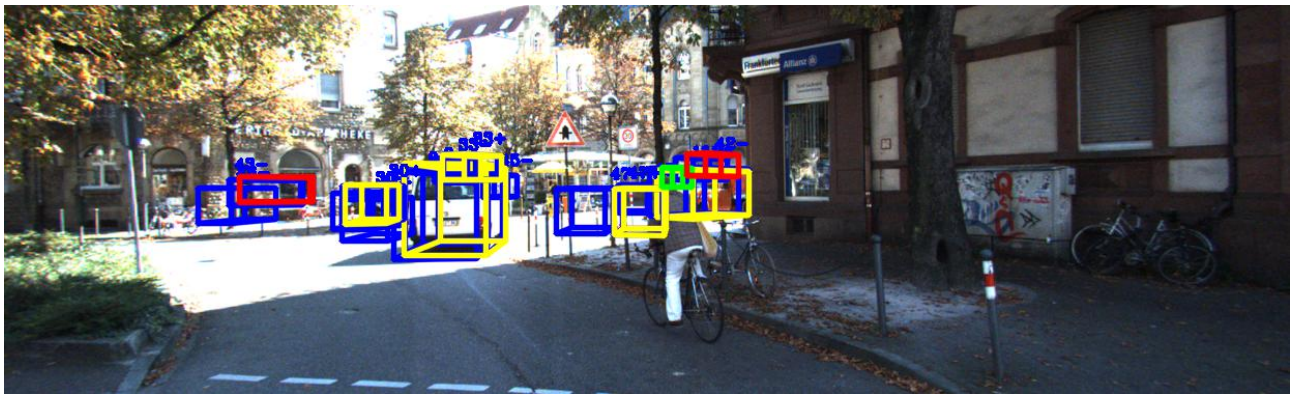
Frame 33



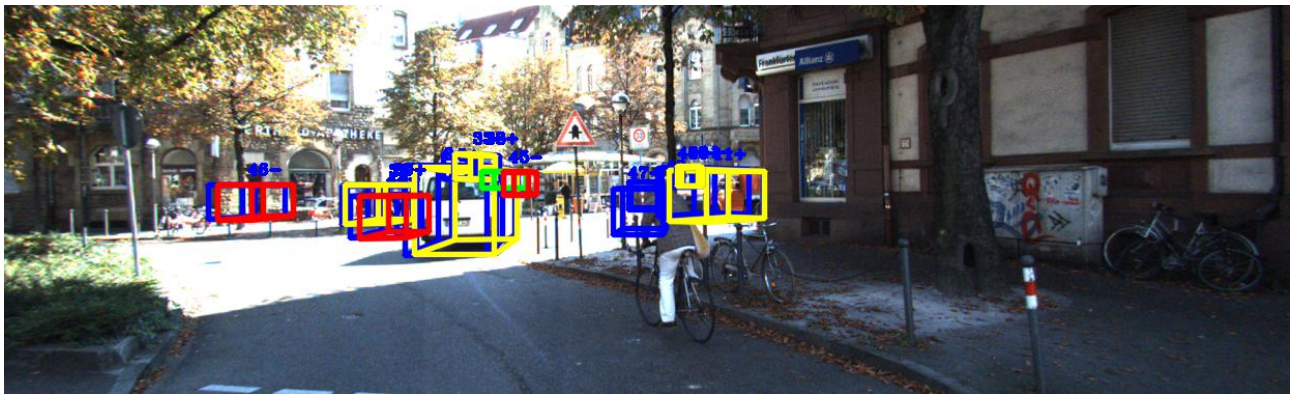
Frame 34



Frame 35



Frame 36



Frame 37

Honestly it's hard to distinguish, but state after prediction is close to detection.

Question 7 (Analysis) [1 pt]: Please run the run `python evaluate.py` and report your MOTA, TPs, ID switches, FRAGs and FPs at the best threshold. Please also visualize at least two failure cases and explain the reason. Please discuss what can be done to avoid these failures.

Question 7 Answer:

```
================evaluation: best results with single threshold================
Multiple Object Tracking Accuracy (MOTA)                              0.8465
Multiple Object Tracking Precision (MOTP)                             0.8345
Multiple Object Tracking Accuracy (MOTAL)                             0.8465
Multiple Object Detection Accuracy (MODA)                             0.8465
Multiple Object Detection Precision (MODP)                            0.9541

Recall                                                                0.9906
Precision                                                             0.9354
F1                                                                    0.9622
False Alarm Rate                                                      0.1883

Mostly Tracked                                                        1.0000
Partly Tracked                                                        0.0000
Mostly Lost                                                           0.0000

True Positives                                                         420
Ignored True Positives                                                 209
False Positives                                                         29
False Negatives                                                          4
Ignored False Negatives                                                111
ID-switches                                                              0
Fragmentations                                                           2

Ground Truth Objects (Total)                                           535
Ignored Ground Truth Objects                                           320
Ground Truth Trajectories                                               12

Tracker Objects (Total)                                                524
Ignored Tracker Objects                                                 75
Tracker Trajectories                                                    44
================================================================================
======================evaluation: average over recall=======================
 sAMOTA  AMOTA  AMOTP
0.9578 0.5677 0.8461
================================================================================
```

$$Q/R = 1$$

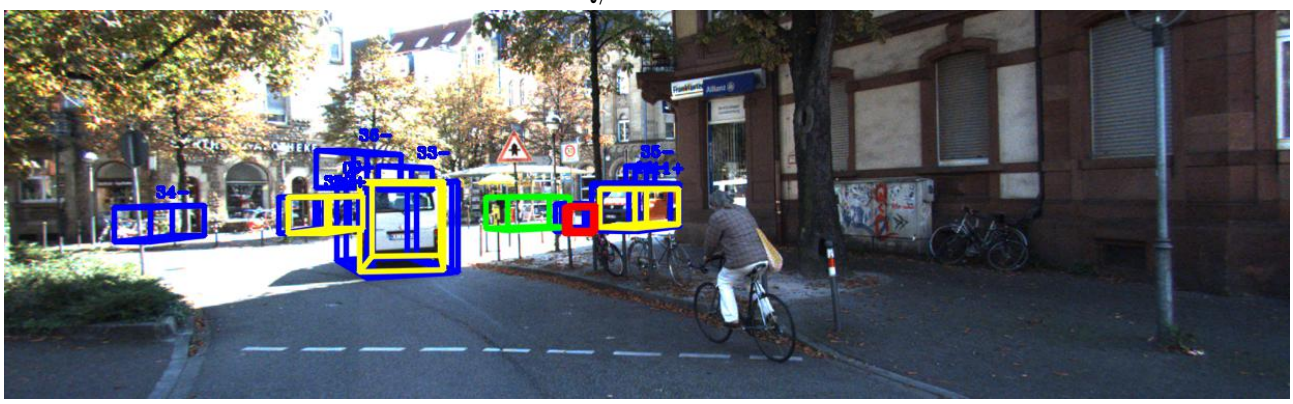
0.5677 is already the best AMOTA I can get (hard to do param search in $\mathbb{R}^{10}$ and $\mathbb{R}^{7}$ even with convexity), change in Σ doesn't make any difference, which is reasonable since it doesn't influence the ratio that we believe more in prediction or measurement.

Failure cases:

1.

```
================evaluation: best results with single threshold================
Multiple Object Tracking Accuracy (MOTA)                              0.8558
Multiple Object Tracking Precision (MOTP)                             0.7502
Multiple Object Tracking Accuracy (MOTAL)                             0.8558
Multiple Object Detection Accuracy (MODA)                             0.8558
Multiple Object Detection Precision (MODP)                            0.9331

Recall                                                                0.9929
Precision                                                             0.9376
F1                                                                    0.9645
False Alarm Rate                                                      0.1818

Mostly Tracked                                                        1.0000
Partly Tracked                                                        0.0000
Mostly Lost                                                           0.0000

True Positives                                                         421
Ignored True Positives                                                 209
False Positives                                                         28
False Negatives                                                          3
Ignored False Negatives                                                111
ID-switches                                                              0
Fragmentations                                                           1

Ground Truth Objects (Total)                                           535
Ignored Ground Truth Objects                                           320
Ground Truth Trajectories                                               12

Tracker Objects (Total)                                                524
Ignored Tracker Objects                                                 75
Tracker Trajectories                                                    46
================================================================================
=======================evaluation: average over recall=======================
 sAMOTA  AMOTA  AMOTP
0.9418 0.5508 0.7675
================================================================================
```

Q/R = 0.01



van disparity

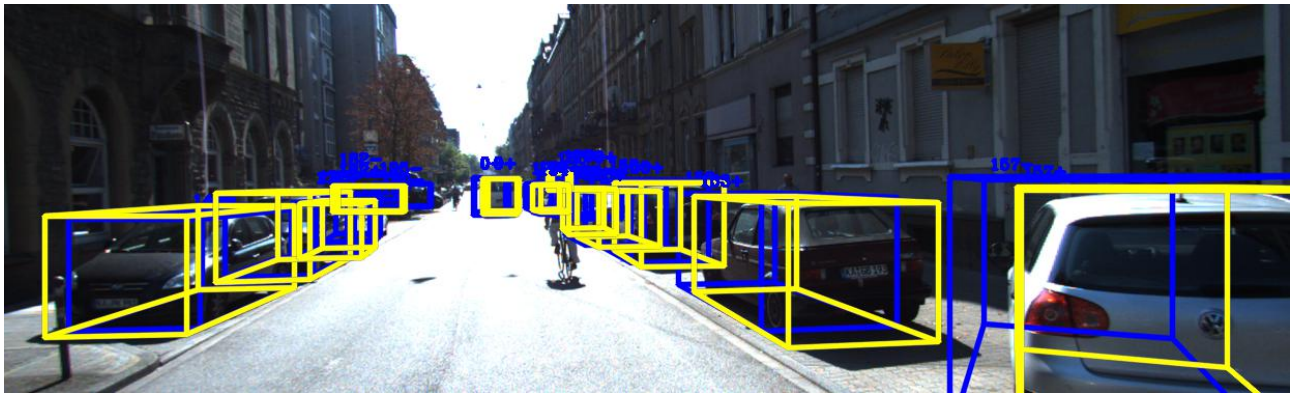
We can see that the state of the van cannot keep up with latest measurement, as the direction has severe disparity.

This can be solved by increasing the uncertainty of prediction/process, or by adding angular velocity as part of the state.

2.

```
================evaluation: best results with single threshold================
Multiple Object Tracking Accuracy (MOTA)                              0.8465
Multiple Object Tracking Precision (MOTP)                             0.8283
Multiple Object Tracking Accuracy (MOTAL)                             0.8465
Multiple Object Detection Accuracy (MODA)                             0.8465
Multiple Object Detection Precision (MODP)                            0.9526

Recall                                                                0.9906
Precision                                                             0.9354
F1                                                                    0.9622
False Alarm Rate                                                      0.1883

Mostly Tracked                                                        1.0000
Partly Tracked                                                        0.0000
Mostly Lost                                                           0.0000

True Positives                                                         420
Ignored True Positives                                                 209
False Positives                                                         29
False Negatives                                                          4
Ignored False Negatives                                                111
ID-switches                                                              0
Fragmentations                                                           2

Ground Truth Objects (Total)                                           535
Ignored Ground Truth Objects                                           320
Ground Truth Trajectories                                               12

Tracker Objects (Total)                                                524
Ignored Tracker Objects                                                 75
Tracker Trajectories                                                    47
================================================================================
======================evaluation: average over recall=======================
 sAMOTA  AMOTA  AMOTP
0.9578 0.5677 0.8393
================================================================================
```

Q/R = 100



state is exactly the same as detection

The other blue boxes indicating trackers after update() is completely occluded by yellow boxes

indicating detections.
The effect is similar to trivial update, and tracker is weak when occlusion happens.
Can be solved by increasing uncertainty of measurement.

Bonus Question (Option 1: Better Matching) [3 pt Bonus]: Improve your matching algorithm from greedy to the Hungarian algorithm. You must implement it yourself. Alternatively improve your matching by training a neural network to perform matching based on more than just spatial proximity. Report the tracking performance and compare it against the IOU-based greedy matching through both qualitative and quantitative results. Do you see a performance improvement? Discuss the comparison study results.

Bonus Question Answer: A partially worked Hungarian algorithm is implemented. I searched through every source I can find about how to propperly program Step 3, which is to find the minimum number of lines covering all zeros(too high level, and embodies most difficulties of this algorithm that I think maybe it's not complete): YouTube Video Tutorial, stackoverflow Question.
Unfortunately non of them are correct. To be short, no existing greedy algorithm can solve this correctly, we need to perhaps use DP, or even brutal force.
Have CS 446 machine learning final on Friday(horrible one since it's pure math), so I couldn't further work on this. ;(
To tell from the first few frames, using Hungarian algorithm doesn't show big advantage in this MP, perhaps because there aren't too many detections and trackers each frame, and many of them are quite apart from each other(thanks to Kalman Filter).

Bonus Question (Option 2:Better Dynamics) [3 pt Bonus]: Make your kalman filter into an extended kalman filter and implement a bycicle model for dynamics instead of linear. `https://www.coursera.org/lecture/intro-self-driving-cars/lesson-2-the-kinematic-bicycle-model-Bi8yE`. Report the tracking performance and compare it against linear velocity model through both qualitative and quantitative results. Do you see a performance improvement? Discuss the comparison study results.