Target Tracking with Kalman Filtering, KNN and LSTMs

Dan Iter, Jonathan Kuck, Philip Zhuang
{daniter, kuck, pzhuang}@stanford.edu

CS229 Machine Learning

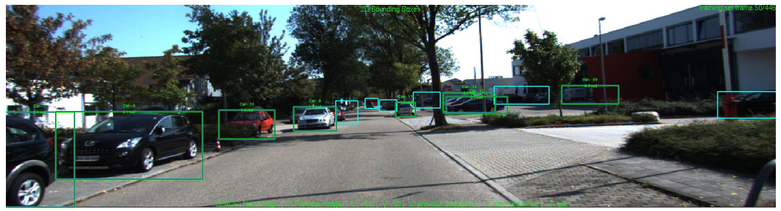


Figure 1: One frame from the KITTI video dataset with ground truth bounding boxes

Prediction Task
Tracking an unknown number of targets given noisy measurements from multiple sensors is critical to autonomous driving. Rao-Blackwellized particle filtering is well suited to this problem. Monte Carlo sampling is used to determine whether measurements are valid, and if so, which targets they originate from. We investigate modeling single target motion using a Kalman filter, K-nearest neighbors, and an LSTM.
We compare single target tracking performance and multi-target tracking performance in the Rao-Blackwellized particle filtering framework.

Data
We test our algorithms on data from the KITTI object tracking benchmark. This dataset is composed of video taken from a car mounted camera while driving around Karlsruhe, Germany. We evaluate single target tracking performance by predicting ground truth bounding box centers which were hand labeled using Mechanical Turk. As multi-target tracking input we use bounding boxes from two separate object detection algorithms, Regionlets and MSCNN. Performance is evaluated using standard multi-target tracking metrics.

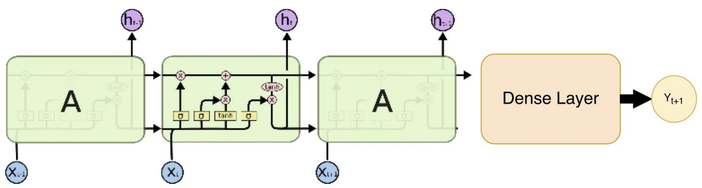
Discussion/Future Work
We demonstrated significant improvement in single target tracking performance using an LSTM instead of a Kalman filter. We found K-nearest neighbors tracking performed comparably to Kalman filtering, but believe we could see improvements given a larger set of training data. Our multi-target tracking experiments have not completed, so we are interested in comparing end to end performance. As future work, we would like to incorporate target dependencies in the LSTM model. Target-measurement association could be improved by utilizing image features.

Kalman Filter
Here is how we configured the Kalman filter. We represented the state variable $X = [x \ v_x \ y \ v_y]^T$ where x, y were coordinates of the center (in pixels) and v_x, v_y were the x, y component of the velocity (in pixels/frame). The initial covariance matrix P follows the variance of X from all training data. The state transition function follows $x_{i+1} = x_i + v_x$ dt and dt = 1 between each frame. The location of valid detections can be viewed as relatively noise free, our measurement noise covariance matrix R was 0 to improve performance. Our measurement only takes x, y coordinates of the object center.
We set the control function B to 0. With the standard Kalman filter predict and update function, we predicted the position of the object in the next frame.

K Nearest Neighbors (KNN)
For each test sequence, we trained KNN on other sequences. For each target in the training set, we recorded every segment of 3 consecutive locations as rows in the training matrix and the next location in the prediction array.
Then, for a given test target, we compared the last three locations to every row in the training matrix and found the 10 nearest neighbors. The predicted location was the weighted average of the next movement of the 10 nearest neighbors as below:

$$(x_{predict}, y_{predict}) = (x_{last}, y_{last}) + \frac{\sum_{i=1}^{10}((x_{predict,i}, y_{predict,i}) - (x_{last,i}, y_{last,i}))(d_{11} - d_i)}{\sum_{i=1}^{10}(d_{11} - di)}$$

where i is the index of the ith nearest neighbor and d_i is the distance between ith nearest neighbor and the test sequence.



Dual LSTMs - Mean and Variance
The LSTM's recurrent structure makes it useful for modeling time series. The figure above demonstrates our architecture. We use a window size of three frames in the past. Having the past three datapoints can be interpreted as containing enough information to capture the location, velocity and acceleration. There is a final dense layer to output the next position predictions.
We also use a similar LSTM to predict the variance in the prediction. Together, the two LSTMs define a gaussian distribution that is used for object association.

Results
We compare the three approaches in their mean squared error for prediction the next position based on the ground truth. We will also evaluate them on their performance in the end to end experiment that uses object detection and object association.

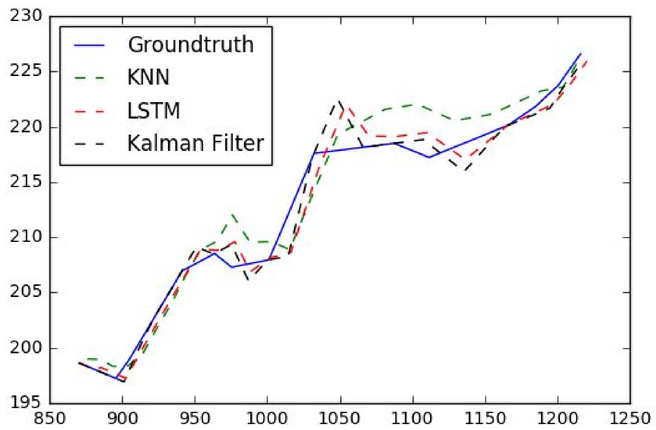


Figure 2: Comparison of different methodologies on one tracking example. x, y axis are coordinates in pixels.

Methodology	Mean Squared Error
Kalman Filter	30.42
KNN	34.79
LSTM	15.14

Table 1. Comparison of MSE between different methodologies

Detections Used	MOTA ↑	MOTP ↑	Mostly Tracked ↑	Mostly Lost ↓
Near-Online (3 frame delay)				
Regionlets Only	78.3% ± .08%	81.4% ± .01%	57.0% ± .49%	8.3% ± .38%
MS-CNN Only	80.9% ± .18%	85.4% ± .01%	69.0% ± .35%	5.5% ± .41%
Regionlets and MS-CNN	**83.3% ± .20%**	84.9% ± .02%	**71.0% ± .54%**	5.1% ± .60%
Online				
Regionlets Only	74.8% ± .14%	81.8% ± .01%	48.2% ± .41%	8.4% ± .37%
MS-CNN Only	78.7% ± .13%	**85.8% ± .01%**	61.9% ± .75%	5.6% ± .44%
Regionlets and MS-CNN	80.4% ± .13%	85.2% ± .02%	65.2% ± .68%	**5.0% ± .27%**

Table 2: Multi-Target Tracking Cross Validation with Kalman Filtering