

Target Tracking with Kalman Filtering, KNN and LSTMs

Dan Iter
daniter@stanford.edu

Jonathan Kuck
kuck@stanford.edu

Philip Zhuang
pzhuang@stanford.edu

December 17, 2016

Abstract

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. This breaks the problem into single target tracking sub-problems that are solved in closed form (e.g. with Kalman filtering). We compare the performance of a traditional Kalman filter with that of a recurrent neural network for single target tracking. We show that LSTMs outperform Kalman filtering for single target prediction by $2\times$. We also present a unique model for training two dependent LSTMs to output a Gaussian distribution for a single target prediction to be used as input to multi-target tracking. We evaluate the end to end performance of an LSTM and a Kalman filter for simultaneous multiple target tracking. In the end to end pipeline, LSTMs do not provide a significant improvement.

1. Introduction

We address the problem of tracking an unknown number of targets given measurements from multiple noisy sensors. Target tracking is a critical problem for autonomous driving. Combining information from different types of sensors (e.g. radar and cameras) is important for reliable and accurate tracking performance in real world settings.

It has been shown that a Rao-Blackwellized particle filter can be used in the multi-target tracking setting [19]. In this framework, Monte Carlo sampling is used to determine whether measurements are valid, and if so, which targets they originate from. This breaks the problem into single target tracking subproblems that are solved in closed form by Kalman filtering.

In the real world cars do not follow linear motion as-

sumptions of the traditional Kalman filter. It is possible to learn non-linear motion models from data using a recurrent neural network [1, 17, 15, 11]. We implement a unique way to train two LSTMs to both predict the future position of a target based on motion and to output a distribution of the prediction's likelihood. The distribution is input into the framework of a Rao-Blackwellized particle filter.

To show that these methods are effective in a real world end to end pipeline, we also incorporate several out of the box methods for object detection (MSCNN and Regionlets). Predicting target motion from noisy measurements output by the object detectors is a critical challenge in this tracking task.

We test our algorithm on the KITTI object tracking benchmark [9]. This dataset is composed of video taken from a car mounted camera while driving around Karlsruhe, Germany. We compare our RNN's location predictions with the naive Kalman filter predictions. Additionally, we incorporate the RNN predictions into a Rao-Blackwellized particle filter to evaluate end to end tracking performance.

2. Related Work

A variety of solutions to the multi-target tracking problem have been presented, including joint probabilistic data association (JPDA) [2], multiple hypothesis tracking (MHT) [2], and finite set statistics (FISST) [14]. The key technical difficulty when tracking multiple targets is determining which target (or clutter) each measurement originated from, referred to as the measurement-target association problem. Under the general framework of multiple hypothesis tracking, probabilities are calculated for every possible combination of measurement-target associations. This quickly becomes intractable, which leads to sequential Monte Carlo approaches. We build on the work of [18, 19], who applied Rao-Blackwellized particle filters to the multiple target tracking problem.

Tracking-by-detection is a common approach when tracking objects in video [5, 3, 12, 16, 6]. Object detection is performed on a frame by frame basis and detec-

tions are then grouped between frames to form target trajectories. The top performing algorithms on the KITTI benchmark follow this strategy [8, 10, 20, 22].

In [8] the authors present a tracking-by-detection based method that is among the top KITTI benchmark performers. The authors use computer vision techniques to improve the association of detections between frames. A local interest point detector and optimal flow algorithm are used to create interest point trajectories. An improved data association method is presented in [10]. The authors propose a new cost function for data association that considers the position, velocity, and size of tracked targets compared with new detections. In [20] the authors associate detections with targets by computing the min-cost network flow for each target in the graph of detections. The model contains quadratic interactions between targets. Model parameters are learned using a structured prediction SVM. Associations between tracked targets and new detections are determined by modelling the state of every target with a Markov decision process (MDP) in [22]. The authors use reinforcement learning to train transition functions in the MDP. Our approaches are more intuitive and interpretable in comparison to many of the above approaches.

3. Dataset and features

We test our method on the KITTI object tracking benchmark [9], a standard for testing the quality of video target tracking algorithms in an autonomous driving setting. The benchmark is composed of video sequences captured by a car mounted camera while driving in and around Karlsruhe, Germany. See an example of a single frame in Figure 1. The benchmark evaluates algorithms using four multi-target tracking metrics, multiple object tracking accuracy (MOTA), multiple object tracking precision (MOTP), the percentage of mostly tracked targets, and the percentage of mostly lost targets. The MOTA and MOTP multi-target tracking metrics were introduced in [4] and have become a standard. The percentage of mostly tracked objects refers to the percentage of ground truth targets that are tracked during more than 80% of their lives while mostly lost targets are tracked for less than 20% of their lifetimes.



Figure 1. This is a single frame from the KITTI dataset. The frame is human annotated with bounding boxes for cars.

We use sets of object detections from two separate algorithms, Regionlets [23, 21, 13] and MS-CNN [7]. The object detections are used as inputs to our tracking algorithm. Also, the KITTI challenge allows for both online and near-online methods. Online methods can not use any data from future frames. Near-only methods can look 3 frames into the future. We present results for both settings.

4. Methods

4.1. Rao-Blackwellized Particle Filter

The multi-target tracking problem can be separated into two subproblems. First is the problem of creating new tracks when a target is born, removing old tracks when a target dies, and associating measurements with either the target they originated from or clutter in the case of false positives. Second is the problem of tracking a single target given only valid measurements that were created by the target. Following this line of reasoning we perform inference in our model using a Rao-Blackwellized particle filter. Target birth, death, and measurement-target association are handled by particle filtering. Then each individual target tracking problem is solved in closed form, which reduces the number of particles needed compared with an approach based solely on particle filtering. We compare target tracking performance when using three different methods to solve the single target tracking problem, a Kalman filter, an LSTM, and a K-nearest neighbors approach.

4.2. Kalman Filter

We used the 2D bounding boxes provided by KITTI’s training sequences as the ground truth. We preprocessed the ground truth bounding boxes and used the sequence of the centers of the bounding boxes of each object as our inputs. Starting from the first center of an object, we predicted the (N+1)th center based on the first N centers using Kalman filtering.

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 components of the velocity (in pixels/frame). With the assumption that variables x , v_x , y and v_y were independent of each other, $P_{ij(i \neq j)} = 0$ in the initial covariance matrix P , and P_{ii} was equal to $\text{var}(X_i)$ from all training data. Since $x_{i+1} = x_i + v_x \times dt$ and between each frame $dt = 1$,

the state transition function $F = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$. We

performed expectation maximization on all training data to find an estimate of the process noise covariance matrix Q . As 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 function $H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$ because only x , y coordinates of the object center were measured. 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.

4.3. KNN

Since cars often follow common patterns enforced by traffic laws, it is reasonable to use K-Nearest Neighbors as a baseline for predicting target motion. Specifically, we find the K most similar patterns that we've seen in our test set and use their weighted average to predict the new location of the target in the next frame. 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:

$$(x_p, y_p) = (x_{last}, y_{last}) + \frac{\sum_{i=1}^{10} ((x_{p,i}, y_{p,i}) - (x_{last,i}, y_{last,i}))(d_{11} - d_i)}{\sum_{i=1}^{10} (d_{11} - d_i)} \quad (1)$$

where (x_p, y_p) is the predicted location, i is the index of the i th nearest neighbor and d_i is the distance between i th nearest neighbor and the test sequence.

4.4. LSTMs

We evaluate LSTMs performance for the single object tracking task. Unlike Kalman Filters, LSTMs make no assumptions about the type of motion of the object, so they should be able to capture both linear and non linear motion. Furthermore, because of the recurrent nature of the neural network, the LSTM can incorpo-

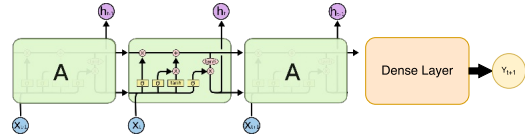


Figure 2. This is the architecture of our 3 window LSTM. It feeds into a dense layer that outputs two floats as the (x,y) coordinate prediction.

rate a window of the previous history when learning to predict the future position of an object. The LSTM learns a regression on the set of $X \rightarrow y$ where X is the observed measurements of object's location in previous frames and y is the prediction of the location of the object in the next frame.

While LSTMs have high capacity, allowing them to capture very complex models, there are a number of downsides to using LSTMs over Kalman Filters. First, LSTMs are a discriminative model, so they output only a predicted value, rather than a distribution. Kalman Filters are generative models and therefore also produce a probability distribution of the prediction which can be useful in other aspects of the tasks (specifically the measurement target association). To combat this issue, we train two LSTMs. One to predict the next position and another to predict the variance in that prediction. Furthermore, LSTMs are much more complex in terms of tuning parameters so it is more challenging to find the optimal model parameters.

. TWO LSTM ARCHITECTURE

The single target prediction task is to predict the position of a particular target on the following frame. The output of the single target prediction is then used in measurement-target association. Therefore, we also need to provide a covariance matrix of the position (x,y) to estimate the likelihood of a target prediction being associated to a measurement. To support this interface, we train two LSTMs.

The first LSTM takes n vectors in \mathbb{R}^2 , which is the last n measurements, where each measurement is an (x,y) coordinate. The second LSTM takes as input the same n vectors, in addition to the prediction output by the first LSTM and outputs a vector in \mathbb{R}^2 which represents the $\text{Var}(X)$ and $\text{Var}(Y)$. Note that we make assume x and y errors are independent. This prevents the maximum likelihood covariance estimate from collapsing to a one dimensional Gaussian.

The LSTM gates recursively feed forward their hidden state and the next window value. The final gate passes the hidden state to a dense layer that outputs the prediction. The window size and hidden layer size

are hyperparameters along with optimization parameters such as the learning rate and number of iterations. Our best results use a hidden layer size of 32, window size of 3 and optimize with default parameters of Adam. Figure 2 visualizes the architecture of the first LSTMs. We hypothesize that the window size of 3 produced good results because 3 points are enough to capture the location, velocity and acceleration of a particle.

To train the first LSTM, we feed in the previous measurements and use the ground truth measurements as the true labels. For the second LSTM, we train on the previous measurements, and the current prediction and optimize for the output to match the maximum likelihood variance in x and y. For a given point, with a prediction \hat{x} and measurement x^* , the maximum likelihood variance in x is $(\hat{x} - x^*)^2$. We compute these variances for each measurement and prediction pair at training and optimize the output of the LSTM to output an equivalent estimate of the variance. The LSTMs are implemented in Keras and are trained with a Theano backend.

. CHALLENGES AND ASSUMPTIONS

A major assumption that we make is that x and y vary independently. Empirically, this is not true. However, by making this assumption we were able to consistently produce sufficient covariance matrices. Also, we train the LSTMs on the ground truth target locations, but at end to end test time, the ground truth target locations are not known. In lieu of the ground truth, we use the measurements output by the object detection framework. These measurements are noisy and therefore increase the variance in the LSTM single object tracking prediction. We compensate for this by approximating the variance in the measurements and adding that variance to the variance output by the second LSTM.

5. Results and Evaluation

5.1. Single Target Prediction

We evaluated the performance of Kalman filters, KNN and LSTMs on single object target prediction and then on the end to end multiple object tracking. For the single target prediction task, we are given the ground truth coordinates in each frame and we compare our predictions to the human annotated ground truth.

Table 2 shows the mean squared error in single target prediction. LSTMs out perform the other methods by 2x. The error is computed by the distance between the predicted coordinate and the ground truth measurement of the target’s location. The models were

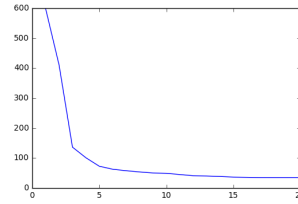


Figure 3. KNN convergence with growth of dataset. The x-axis is the number of training videos used. The y-axis is the mean squared error.

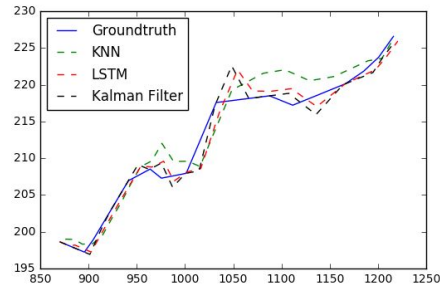


Figure 4. This figure shows the path of a single object. The other lines represent the predictions made by our trained Kalman filter, KNN and LSTM models.

trained on ground truth and were evaluated using 21 fold cross validation (because there were 21 separate contiguous videos).

While KNN was used as a baseline, it revealed a number of key characteristics about the data. Figure 3 show the mean squared error of KNN based on the number of videos that were used for training. This shows that there are many very common patterns that are basically captured entirely over the course of 5 videos. However, then the rate of improvement quickly drops and the line flattens out. This indicates that most of the additional patterns seen are mostly exceptional and are not particularly useful for prediction. It is also worth noting that KNN was substantially slower at test time than the other methods.

Figure 4 shows the path of a single object over time and the predictions made by each of the methods we evaluate. This is just a single example but it gives some insight as to how well each method tracks the object. In particular notice how predictions overshoot when the object turns and how it may take some time to readjust to the new trajectory.

5.2. Multi-Target Prediction and Association

In the end to end tracking task, we use the prediction and object detections as input to the Rao-Blackwellized particle filter to associate each predic-

Cross Validation on KITTI Training Data

Detections Used	MOTA \uparrow	MOTP \uparrow	Mostly Tracked \uparrow	Mostly Lost \downarrow
Kalman Filter: Near-Online (3 frame delay)				
Regionlets Only	78.3% \pm .08%	81.4% \pm .01%	57.0% \pm .49%	8.3% \pm .38%
MS-CNN Only	80.9% \pm .18%	85.4% \pm .01%	69.0% \pm .35%	5.5% \pm .41%
Regionlets and MS-CNN	83.3% \pm .20%	84.9% \pm .02%	71.0% \pm .54%	5.1% \pm .60%
Kalman Filter: Online				
Regionlets Only	74.8% \pm .14%	81.8% \pm .01%	48.2% \pm .41%	8.4% \pm .37%
MS-CNN Only	78.7% \pm .13%	85.8% \pm .01%	61.9% \pm .75%	5.6% \pm .44%
Regionlets and MS-CNN	80.4% \pm .13%	85.2% \pm .02%	65.2% \pm .68%	5.0% \pm .27%
KNN: Near-Online (3 frame delay)				
Regionlets Only	70.0%	81.5%	51.7%	10.4%
LSTMs: Near-Online (3 frame delay)				
Regionlets Only	76.4%	81.5%	54.1%	9.3%
MS-CNN Only	81.23%	85.6%	66.3%	4.6%
Regionlets and MS-CNN	82.6%	85.0%	70.5%	5.3%

Table 1. We performed cross validation on KITTI training data for the fully online version of our algorithm and with a three frame delay. Performance of both versions of our algorithm are given using only Regionlets detections, only MS-CNN detections, and both detection sources together. Values are given as the (mean \pm 1 standard deviation) over 10 runs.

Methods	Mean Squared Error
Kalman Filter	30.42
KNN	34.79
LSTM	15.14

Table 2. Comparison of MSE between different methodologies

tion with an object. Cross validation results on the KITTI target tracking dataset are shown in Table 2.

Despite the large improvement in single target tracking using the LSTMs, the end to end performance is similar for the LSTM and the Kalman filter. This is likely because target association is based on both the prediction and the distribution of where predictions could possibly associate to measurements. If the second LSTM fails to provide good variance estimates, the end to end model will incorrectly predict new objects appearing or objects disappearing instead of correctly associating them to a measurement. The MOTA evaluation metric incorporates false positive, missed target, and id-switch counts. While the overall MOTA is similar when using a Kalman filter or LSTM to predict target motion, the number of target id-switches is reduced when using an LSTM. In the case of using MS-CNN detections with a 3 frame delay, replacing the Kalman filter with an LSTM reduces the number of id-switches from 232 to 157. In the current framework we represent the likelihood of target birth and measurement association with clutter

using uniform distributions. It is possible that using a more descriptive distribution could improve the overall MOTA when using an LSTM to predict motion.

6. Future work

This work focused on learning to track targets by their motion, completely disregarding visual features. We can incorporate characteristics from the image, such as car color, to improve measurement target association.

Also, the model makes many simplifying assumptions that hurt our end to end performance. For example, the probability of a target appearing is uniform over the image. If we remove this assumption, we may be able to better distinguish between noisy predictions and targets disappearing or appearing in the view.

7. Conclusion

Target tracking for autonomous vehicles is a challenging task with many components, each of which is its own machine learning task. In this work, we were able to break the task into a pipeline of smaller problems. We explored LSTMs, Kalman filters and KNN for single target motion prediction, Rao-Blackwellized particle filtering for measurement target association, and several prepackaged techniques for object detection. We evaluated many combinations of pipelining these techniques and presented several feasible methods for building a multi-target tracking system.

References

- [1] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces.
- [2] Yaakov Bar-Shalom and Xiao-Rong Li. Multitarget-multisensor tracking: principles and techniques. *Storrs, CT: University of Connecticut, 1995.*, 1995.
- [3] Jerome Berclaz, Francois Fleuret, Engin Turetken, and Pascal Fua. Multiple object tracking using k-shortest paths optimization. *IEEE transactions on pattern analysis and machine intelligence*, 33(9):1806–1819, 2011.
- [4] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008(1):1–10, 2008.
- [5] Michael D Breitenstein, Fabian Reichlin, Bastian Leibe, Esther Koller-Meier, and Luc Van Gool. Robust tracking-by-detection using a detector confidence particle filter. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1515–1522. IEEE, 2009.
- [6] Michael D Breitenstein, Fabian Reichlin, Bastian Leibe, Esther Koller-Meier, and Luc Van Gool. Online multiperson tracking-by-detection from a single, uncalibrated camera. *IEEE transactions on pattern analysis and machine intelligence*, 33(9):1820–1833, 2011.
- [7] Zhaowei Cai, Quanfu Fan, Rogerio S Feris, and Nuno Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. *arXiv preprint arXiv:1607.07155*, 2016.
- [8] Wongun Choi. Near-online multi-target tracking with aggregated local flow descriptor. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3029–3037, 2015.
- [9] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.
- [10] Ju Hong Yoon, Chang-Ryeol Lee, Ming-Hsuan Yang, and Kuk-Jin Yoon. Online multi-object tracking via structural constraint event aggregation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1392–1400, 2016.
- [11] Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. *CVPR*, 2016.
- [12] Cheng-Hao Kuo, Chang Huang, and Ramakant Nevatia. Multi-target tracking by on-line learned discriminative appearance models. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 685–692. IEEE, 2010.
- [13] Chengjiang Long, Xiaoyu Wang, Gang Hua, Ming Yang, and Yuanqing Lin. Accurate object detection with location relaxation and regionlets re-localization. In *Asian Conference on Computer Vision*, pages 260–275. Springer, 2014.
- [14] Ronald Mahler. Random sets: Unification and computation for information fusion—a retrospective assessment. In *Proceedings of the Seventh International Conference on Information Fusion*, volume 1, pages 1–20. I, 2004.
- [15] Anton Milan, Seyed Hamid Rezaatofighi, Anthony Dick, Konrad Schindler, and Ian Reid. Online multi-target tracking using recurrent neural networks. *arXiv preprint arXiv:1604.03635*, 2016.
- [16] Anton Milan, Stefan Roth, and Konrad Schindler. Continuous energy minimization for multitarget tracking. *IEEE transactions on pattern analysis and machine intelligence*, 36(1):58–72, 2014.
- [17] Peter Ondruska and Ingmar Posner. Deep tracking: Seeing beyond seeing using recurrent neural networks. *arXiv preprint arXiv:1602.00991*, 2016.
- [18] Simo Särkkä, Aki Vehtari, and Jouko Lampinen. Rao-blackwellized monte carlo data association for multiple target tracking. In *Proceedings of the seventh international conference on information fusion*, volume 1, pages 583–590. I, 2004.
- [19] Simo Särkkä, Aki Vehtari, and Jouko Lampinen. Rao-blackwellized particle filter for multiple target tracking. *Information Fusion*, 8(1):2–15, 2007.
- [20] Shaofei Wang and Charless Fowlkes. Learning optimal parameters for multi-target tracking. In *British Machine Vision Conference*, 2015.
- [21] Xiaoyu Wang, Ming Yang, Shenghuo Zhu, and Yuanqing Lin. Regionlets for generic object detection. *IEEE transactions on pattern analysis and machine intelligence*, 37(10):2071–2084, 2015.
- [22] Yu Xiang, Alexandre Alahi, and Silvio Savarese. Learning to track: Online multi-object tracking by decision making. In *Proceedings of the IEEE*

International Conference on Computer Vision,
pages 4705–4713, 2015.

- [23] Will Y Zou, Xiaoyu Wang, Miao Sun, and Yuanqing Lin. Generic object detection with dense neural patterns and regionlets. *arXiv preprint arXiv:1404.4316*, 2014.