Deep Tracking on the Move: Learning to Track the World from a Moving Vehicle using Recurrent Neural Networks

Julie Dequaire, Dushyant Rao, Peter Ondrúška, Dominic Zeng Wang and Ingmar Posner

Abstract— This paper presents an end-to-end approach for tracking static and dynamic objects for an autonomous vehicle driving through crowded urban environments. Unlike traditional approaches to tracking, this method is learned end-to-end, and is able to directly predict a full unoccluded occupancy grid map from raw laser input data. Inspired by the recently presented *DeepTracking* approach ([1], [2]), we employ a recurrent neural network (RNN) to capture the temporal evolution of the state of the environment, and propose to use *Spatial Transformer* modules to exploit estimates of the egomotion of the vehicle. Our results demonstrate the ability to track a range of objects, including cars, buses, pedestrians, and cyclists through occlusion, from both moving and stationary platforms, using a single learned model. Experimental results demonstrate that the model can also predict the future states of objects from current inputs, with greater accuracy than previous work.

I. INTRODUCTION

The safe and effective operation of an autonomous vehicle depends on its ability to interpret its surroundings and track and predict the state of the environment over time. Many tracking systems employ multiple hand-engineered stages (e.g. object detection, semantic classification, data association, state estimation and motion modelling, occupancy grid generation) in order to represent the state and evolution of the world ([3], [4], [5]). However, as the tasks assigned to robots become more complex, this approach becomes increasingly infeasible.

Recent advances in machine learning, particularly those of deep neural networks, have demonstrated the ability to capture complex structure in the real world, and have led to significant improvements in numerous computer vision and natural language processing applications ([6], [7], [8]). Such approaches would however typically require large, task-specific corpora of annotated ground-truth labels to master the desired task. This becomes difficult when learning a model of the environment without access to corresponding ground truth, as is often the case for object tracking in crowded urban environments.

In recent work, [2] took an alternative approach and presented an end-to-end fully and efficiently trainable framework for learning a model of the world dynamics, building on the original *DeepTracking* work by [1]. We considered the specific problem of learning to track and classify moving objects in a complex and only partially-observable real-world scenario, as viewed from a *static* sensor. Here, we

Authors are from the Mobile Robotics Group at the University of Oxford, United Kingdom: `julie, dushyant, ingmar@robots.ox.ac.uk`
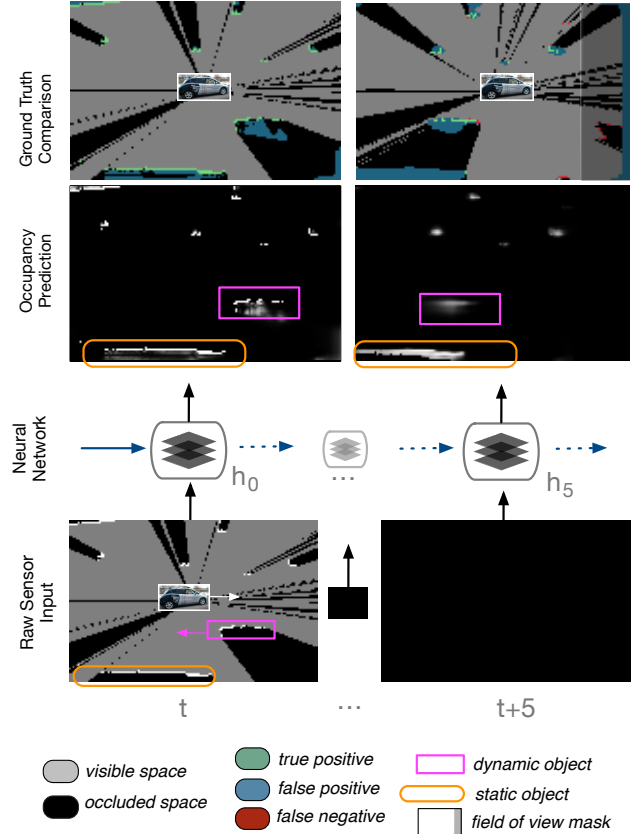
Fig. 1. A typical training sequence. The unoccluded occupancy map is predicted directly from the input grid data, allowing objects to be tracked in occlusion and in future frames. The observed false positives are therefore beneficial. Comparison to the visible ground truth shows that the model is able to capture the dynamics of the moving vehicle (pink rectangle) and accurately predicts its track.

advance this work and address the problem of tracking from a *moving* platform. We extend the neural network architecture proposed in [2] to account for the egomotion of the sensor frame as it moves in the world frame, and demonstrate improved tracking accuracy as compared to previous work.

We demonstrate the system on laser point cloud data collected in a busy urban environment, with an array of static and dynamic objects, including cars, buses, pedestrians and cyclists. The model not only bypasses the sequence of hand-engineered steps typical of traditional tracking approaches, but is empirically shown to successfully predict the future evolution of objects in the environment, even when they are completely occluded.

The rest of the paper is structured as follows. Section II

highlights related work and Section III summarises the problem definition and *DeepTracking* framework first presented in ([1], [2]). Section IV describes the models used to perform tracking in real-world scenarios considering both static and dynamic sensors. Section V presents an empirical evaluation of our methods, and Section VI concludes the paper and discusses the future implications of our findings.

II. RELATED WORK

A number of previous works have explored the problem of model-free tracking of objects in the environment of an autonomous vehicle ([3], [4], [5]). Typically, these approaches follow the traditional paradigm of a multi-component pipeline, with separate components to parametrise and detect objects, associate new measurements to existing tracks, and estimate the state of each individually tracked object. The use of multiple stages in the framework is cumbersome and introduces extra unnecessary failure modes for the tracking algorithm.

Recent work proposes to replace these multiple stages with an end-to-end learning framework known as *DeepTracking*, by leveraging neural networks to directly learn the mapping from raw laser input to an unoccluded occupancy grid ([1], [2]), even with relatively small amounts of data. The approach utilises an RNN architecture using *gated recurrent units* [9] to capture the state and evolution of the world in a sequence of laser scan frames. Another work [10] considers *recurrent flow networks* and takes a different angle to predicting occupancy in dynamic environments. They explicitly encode a range of velocities in the hidden layers of a recurrent network, and use Bayesian optimization to learn the network parameters which update velocity estimation and occupancy prediction. However, the model does not explicitly track objects through occlusion.

DeepTracking shares similarities with deep learning approaches to predictive video modelling ([11], [12]), in that it is trained to predict the future state of the world based on current input data. This is particularly important, because in order to successfully capture the future location of dynamic objects in the scene, the model must implicitly store the position and velocity of each object in its internal memory state.

While this eliminates the need to design individual components by hand, the model assumes a static vehicle [2]. Extending the problem to a moving platform is a challenging task, as it introduces an array of complex relative motions between the vehicle and objects in its environment. As *DeepTracking* ignores the motion of the vehicle, the model is forced to learn all possible motion interactions between the vehicle and its environment as if the vehicle were stationary. For a moving platform, we leverage estimates of egomotion as a proxy for vehicle motion. We scale up the RNN-based models proposed by ([1], [2]) for real-world application on dynamic vehicles, and exploit *Spatial Transformer* modules [13], which allow the internal memory state representations to be spatially transformed according to the estimated egomotion.

The main contributions of this work are as follows:
1) A more in-depth analysis of the performance of *DeepTracking* in the case of a static vehicle, building on the experiments presented in [2].
2) The use of *Spatial Transformer* modules to exploit estimates of visual egomotion in the *DeepTracking* framework.
3) Demonstration of end-to-end tracking of a variety of object classes through occlusion, on a moving vehicle in crowded urban scenes.

III. TRACKING PROBLEM FORMULATION

The problem we address in this paper is to uncover the true state of the world, in terms of a 2D occupancy grid y_t, given a sequence of *partially observed* states of the environment $x_{1:t}$ computed from raw sensor measurements. In particular, we solve for $P(y_t|x_{1:t})$, the probability of the true unoccluded state of the world at time t given a sequence of partial observations at all previous time steps. This formulation can also be used to predict future states by solving for $P(y_{t+n}|x_{1:t})$, given empty input for $x_{t+1:t+n}$.

$$P(y_t|x_{1:t}) = P(y_t|h_t) \qquad (1)$$

Evolution of this latent state h_t, which includes propagating model dynamics and integrating new sensor measurements, is modelled by the update equation:

$$h_t = f(h_{t-1}, x_t) \qquad (2)$$

The key element here is that both the latent state update $f(h_{t-1}, x_t)$, and the decoding step to produce the output $P(y_t|h_t)$ are modelled as parts of a single neural network and are trained jointly. Equations 1 and 2 can then be performed repeatedly as the building block of a recurrent neural network that continuously updates the belief state h_t, and uses it as network memory to predict y_t. This makes it suitable for online stream filtering of sensor input.

When the output ground-truth y_t is not readily available, as is the case in real-world scenarios, the network can be trained in an self-supervised fashion. This is made possible by considering that predicting the movement of objects in occlusion at time t is similar to predicting a *future* state y_{t+n} provided no input is given to the network, i.e $x_{t+n} = \varnothing$. Lack of input observation equates to complete occlusion of the scene. As only *observable* ground truth is available, we reduce the problem of predicting y_{t+n} to that of predicting the directly observable part of y_{t+n}, denoted y'_{t+n}, which equates to the observed input x_{t+n}. Training the network to instead predict $P(y'_{t+n}|x_{1:t})$ is equivalent to computing and backpropagating the errors only on the observable parts of the scene. We refer the reader to [1] and [2] for further details on the RNN and the training procedure.

Each input observation $x_t \in \{0,1\}^{2 \times M \times M}$ is represented as a pair of discretised 2D binary grids of size $M \times M$, parallel to the ground and locally built around the sensor. The first matrix encodes whether a cell is directly observable by the sensor at time t, while the second matrix encodes whether
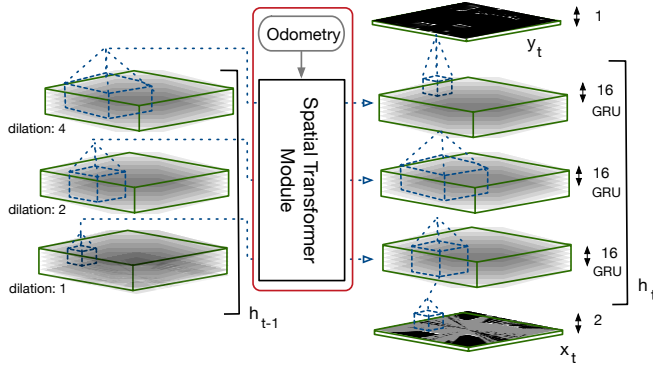
Fig. 2. The proposed network architecture features dilated convolutions, gated recurrent units, a spatial transformer module, and outputs cell occupancy in the sensor's surroundings. The spatial transformer module is only utilised in a moving vehicle scenario.

a cell is observed to be free (value of 0) or occupied (value of 1). We refer to these two matrices as $x_{t,vis}$ and $x_{t,occ}$, the *visibility* and *occupancy* grids respectively. The output we wish to obtain is an occlusion-free state of the world $y_t \in \{0,1\}^{M \times M}$, and is represented by an occupancy matrix similar to that of the input occupancy grid of x_t.

In the next section, we build upon [2] to deploy the *DeepTracking* paradigm on a real-world moving platform.

IV. TECHNICAL OVERVIEW

A. Deep Tracking from a Static Sensor

First, we revisit [2] to detail the baseline *DeepTracking* architecture for real world application, which we extend to a moving platform.

At each time step t, the partially observed grid x_t used as input to the network is computed from raw 2D laser scans by ray tracing. Cells in which a measurement ends are marked as occupied, all cells from the sensor origin up to the end of the ray are marked as free, and cells beyond the ray are marked to be unobserved.

The input x_t is processed by a multi-layer network illustrated in Figure 2, with the Spatial Transformer module only utilised in the moving vehicle scenario. The architecture originally proposed in [1] is scaled up with dilated convolutions [14] and a variant of gated recurrent units ([9], [15]) to allow for the simultaneous tracking of different-sized objects such as cars and pedestrians. Each layer l at time t is updated considering its own activations at time $t-1$ and those of layer $l-1$ below at time t, thus implementing the recurrence. This allows the network to extract and remember the information from the past and to use it for prediction of occluded objects or future states of the world. An additional static memory can also be utilised, in which the network is able to learn individual pixel-wise biases to add to the output of every convolutional operation. The output of the final layer is then converted into the resulting output of cell occupancy y_t via a simple convolutional decoder.

B. Deep Tracking from a Moving Vehicle

Tracking a dynamic scene from a moving platform poses the challenge of decoupling the motion of the vehicle from the motion of objects in the global environment. In the static scenario, information related to an object located at index $\{i, j\}$ in the input x_t would typically be stored at the corresponding neighbouring spatial location $\{i+\Delta i, j+\Delta j\}$ in the layers of the latent representation, with the neighbourhood $\Delta_{i,j}$ based on the receptive field of each neuron in the hidden state. The latent state update would then pass the information along spatially in accordance with the observed relative motion of the input object (Figure 4). In the static scenario, the dynamics of the scene as viewed from the world frame are coherent with that viewed from the local sensor frame.

When tracking from a moving vehicle however, the spatial update of information within the latent representation would additionally need to account for the sensor's egomotion as it will affect the position of the object relative to the sensor frame. Although this is a major drawback of the baseline *DeepTracking* architecture, it can be compensated by transforming the memory state in accordance to the egomotion. In other words, a static obstacle situated at position $\{i_{t-1}, j_{t-1}\}$ in the sensor frame at $t-1$, will be moved to position $\{i_t, j_t\}$ in the sensor frame at time t such that:

$$[x_t, y_t, 1]^T = T_{t,t-1} \times [x_{t-1}, y_{t-1}, 1]^T \qquad (3)$$

where $T_{t,t-1}$ is the $\mathbb{SE}(2)$ forward transformation of the sensor source frame at $t-1$ into the sensor destination frame at t. This formulation naturally extends to 3D motion with $\mathbb{SE}(3)$.

We aid the network in decoupling egomotion and object motion in this way by introducing a *Spatial Transformer* [13] module (STM) in the hidden state (Figure 2). In the original work by [13], the STM is introduced as a learnable module for actively spatially transforming feature maps to aid tasks such as the classification of distorted datasets. However, in the context of tracking from a moving platform where egomotion is readily available (e.g. visual odometry), the STM can be used just to forward transform the hidden feature maps centred in the sensor source frame at time $t - 1$, into the sensor destination frame at time t, using transformation $T_{t,t-1}$ (Equation 3).

Thus, the STM is introduced in the hidden state and performs a transformation on all feature maps of h_{t-1}, set in frame $t - 1$, into frame t before update with the new incoming input x_t to form the new memory at t.

C. Training

In both static and dynamic cases, the network is presented the input sequence $x_{1:t}$ and trained to predict the next n input frames $x_{t+1:t+n}$. The binary cross-entropy loss is calculated and backpropagated only on the visible part of the output, which is achieved by simply masking the prediction y_t with $x_{t,vis}$ and multiplying the resulting grid element-wise with the occupancy part grid $x_{t,occ}$. By using a loss that encourages the model to predict the state of the environment in the future, the model is forced to capture the motion of each object in the hidden representation.

In the static sensor scenario, the error is only backpropagated on the ground truth available, i.e the *visible* part of the space. In the case of a moving sensor, however, an additional constraint needs to be imposed to account for the fact that as the robot moves in future frames, it will discover new space that falls outside the field of view of the current frame. Given this fact, the model should not be falsely penalised for failing to accurately guess objects located within this new space when the input is blanked out. This is similar in nature to the static case, where the input grid also represents a frontier between what the robot can perceive and understand of the scene and the unknown world outside its field of view.

To address this, we apply an additional mask at training time on the cost computation and error backpropagation to represent the *predictable* space in future frames. Accounting for this field of view drift is crucial in terms of tracking performance, as it corrects for an objective function that is otherwise skewed towards the incredibly difficult task of predicting objects outside the field of view. This mask has been overlaid in transparency over the ground truth comparison outputs of Figure 1, and indicates the predictable free space shrinking on future timesteps.

V. EXPERIMENTAL RESULTS

In this section, we perform experimental validation of both the baseline *DeepTracking* and the STM-based variant proposed in this paper. With a stationary vehicle, no spatial transform is necessary and the two are identical.

A. Static Vehicle

For the static case, we consider an architecture with three hidden layers of 16 GRU feature maps each. Computation of the hidden state consists of 3×3 convolutional filters, applied on the three layers (from bottom up) with strides of 1, 2, and 4 pixels, and receptive fields of 3×3, 7×7, and 11×11, respectively. We consider an input field of view of 20×20 m^2 discretised into cells of 20×20 cm^2, which translates into a $H \times W = 101 \times 101$ input grid. With a hidden state consisting of 48 feature maps, the additional static memory contributes to $H \times W \times D = 489,648$ of the total $1,506,865$ parameters of the network.

The evaluation dataset consists of a 75 minute log collected from a stationary *Hokuyo* UMT-30LX 2D laser scanner, positioned in the middle of a busy urban intersection. The area features dense traffic composed of buses, cars, cyclists and pedestrians, which results in extensive amounts of occlusion. Consequently, at no point in time is the complete scene fully observable. The dataset is subsampled at 8Hz and split into a 65 minute set for training, and a 10 minute set for testing occupancy prediction.

The full model was trained for 100 epochs over three days on an Nvidia Tesla K40 GPU, using the unsupervised training procedure described in Section III. The training set is split into mini-batch sequences of 40 frames (5 seconds). For every mini-batch, the network is shown 10 frames and trained to predict the next 10 frames, leading to two such sequences per 40-frame mini-batch. This length of sequence
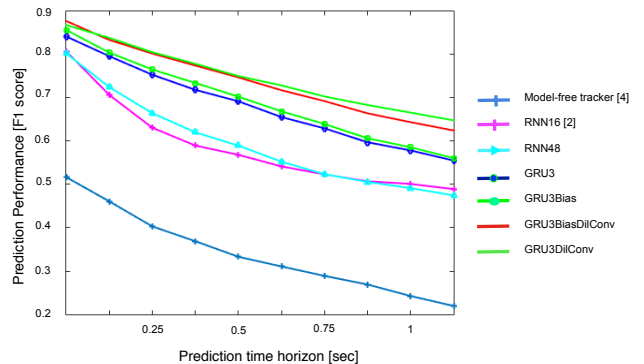

Fig. 3. F1 scores of network architectures when attempting to predict the future occupancy of the scene in a 1.25 second time horizon. The F1 measure is computed with a threshold of 0.5 when considering a cell to be predicted as occupied or free.

covers the typical lengths of occlusions observed but can be tuned accordingly.

1) Quantitative results: We first look to quantify the gain in performance achieved by scaling up the original *DeepTracking* network with the proposed architecture. We compare a number of different architectures ranging from the original [1] to the proposed model in [2], and compare performance on the task of predicting the observable near-future scene occupancy $y'_{t+1:t+n}$, given the input sequence $x_{1:t}$. The predicted output occupancy $P(y_{t+n}|x_{1:t})$ is compared to $x_{t+n,vis} = y'_{t+n}$, which corresponds to the visible (hence known) part of the world at time t. A threshold of 0.5 is used to determine whether a cell is predicted as occupied or free, and an F1 measure computed for each frame.

Figure 3 compares the F1 measures computed on each of $n = 10$ blacked out future frames, given the 10 frames in the past. All model predictions decrease over time as would be expected, as the uncertainty of the state of the world increases with the prediction horizon. There is a notable step change in performance with neural architectures compared to a state-of-the art model-free tracking pipeline approach by [4]. If we increase the capacity of the original RNN architecture of [1] (RNN16) from 16 to 48 feature maps (RNN48), we do not find any significant performance increase. In comparison, replacing the standard RNN hidden state with the modified *gated recurrent units* (GRU3) detailed in [2] provides significant improvement in prediction ability. Finally, although incorporating static memory capacity (GRU3Bias) does yield some improvement over GRU3 alone, it is the dilated convolutions (GRU3DilConv) that, in concert with GRUs in the hidden state, yields the highest performance with significant margin. The performance of the full model (GRU3BiasDilConv), however, is commensurate to that of GRU3Dil, and the learned static bias values may convey useful information such as that of the static background layout.

2) Qualitative results: To better understand what the network has learned, we also qualitatively analyse a typical test sequence of length 3 seconds, along with the network output and selected hidden state feature maps in Figure 4. The network is able to track pedestrians through full occlusion
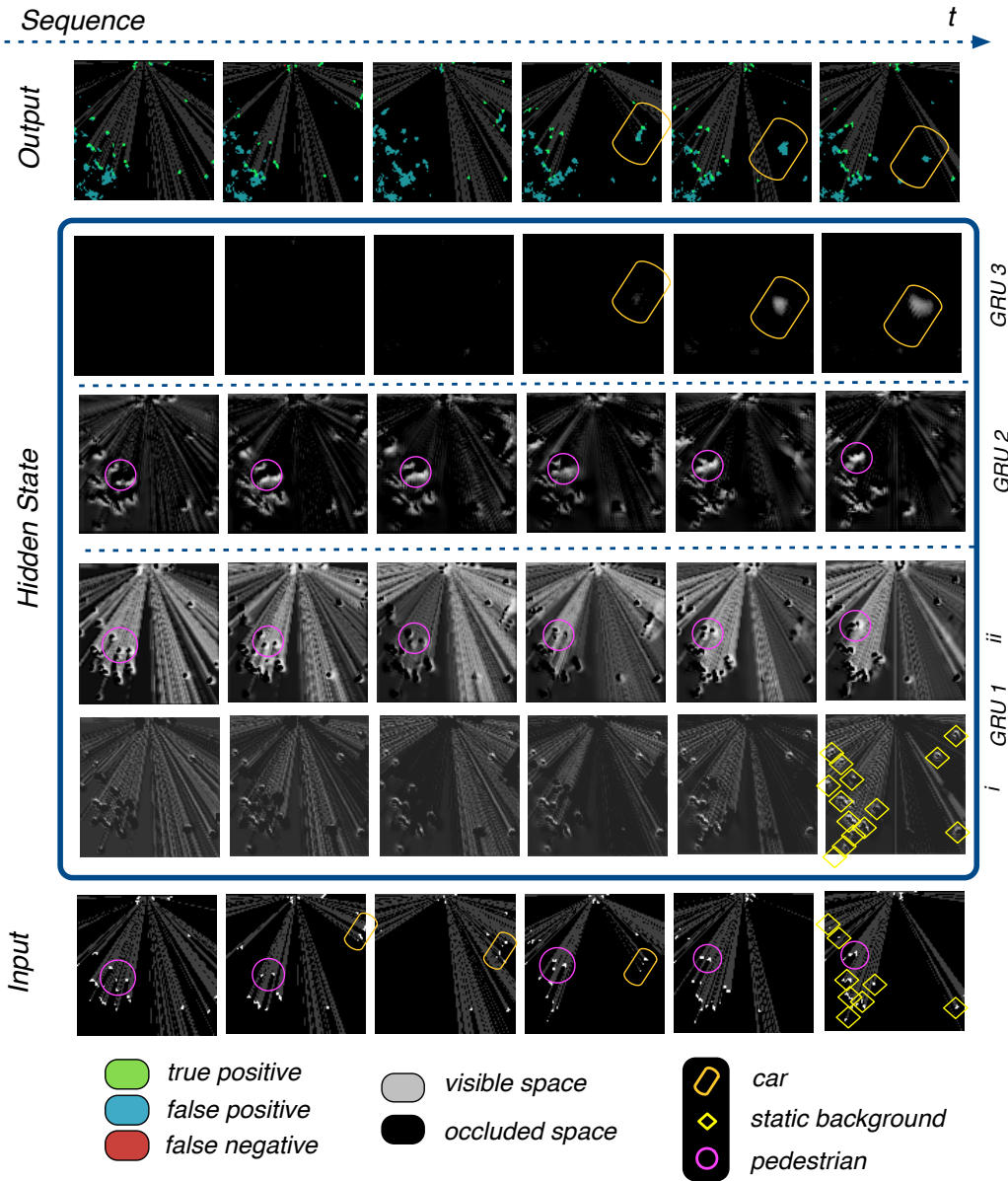
Fig. 4. Example of outputs produced by the system along with a selection of activations in the hidden layers. As highlighted in colour-coded circles, the network is able to propagate the assumed motion of the objects even when in complete occlusion. The sample hidden layer activations shown highlight the fact that lower layers in the hidden units (corresponding to a low dilation of the convolutions) capture the motion of small and slow moving objects such as pedestrians (e.g pink circle) and static background (e.g yellow squares), whereas a higher level layer learns to detect moving vehicles (orange oval).

and the unobserved hallucinated tracks are represented in blue in the output sequence.

It can be seen that the GRU1 feature maps appear to have learned to capture the static background, as activations remain stationary during the sequence (i), and track pedestrians, as highlighted through the pink circles (ii). The GRU2 feature map also captures the motion of pedestrians moving upwards to the left, while, interestingly, the GRU3 unit seems to activate only on the car that appears to the top right at frame 2 (indicated by the orange box). This provides empirical support for the use of Dilated Convolutions, as they allow the model to learn patterns from a wider receptive field.

In general, we observe that information regarding objects in the scene is captured in the hidden state, and moves spatially through the feature maps according to the motion of the object in the input. This can be problematic when extending tracking to a moving platform, as the object motion and vehicle motion are coupled. We address this concern in the following section.

B. Moving vehicle

In this section we compare the best performing model from the static case, which utilises GRU3 and Dilated Convolutions (Baseline DT), with an equivalent architecture that incorporates the Spatial Transformer module into the hidden belief state (STM). As with the static case, we evaluate the ability to predict future frames given blacked out input, and illustrate the achieved occlusion-free tracking performance on a series of examples selected from the test set.
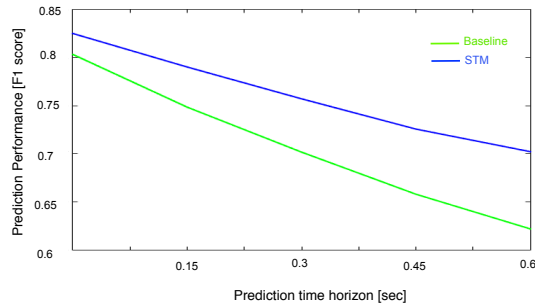
Fig. 5. Positive contribution of the Spatial Transformer to the network's ability to correctly predict the future occupancy of the scene in a 0.6 sec time horizon. The baseline DT does surprisingly well which we attribute to the benign test set where the vehicle mostly evolves at constant velocity down straight roads.

The evaluation dataset was collected over a 35 minute period, from a moving vehicle equipped with two Velodyne HDL64E lasers, resulting in a 360 degree field of view. The 3D point clouds were reduced to a 2D scan by considering the range of points within 0.6-1.5 meters height from the ground.

The network was trained on mini-batches of 40 sequences with a frame rate of 10Hz, alternating between 5 inputs shown, and 5 inputs hidden. This higher frequency is better adapted to the moving vehicle case given the input field of view of 18×18 m^2 and a vehicle mean velocity of 20 miles per hour. Longer occluded sequences would lead to an increase loss in useful memory due to the aforementioned drift of field of view. Finally, we performed an 80/20 split of the data into training and test set with no overlap in location, and trained our GRU3DilConv model for 100 epochs over 2 days on an Nvidia Tesla K40 GPU until convergence. Our architecture is implemented on Torch and uses the *Spatial Transformer* GPU-implementation of [16].

a) Quantitative Results: Figure 5 represents the F1 measure as computed with STM and with the baseline DT. The baseline DT is identical to STM with the exception that no egomotion is taken into account. In other words, no additional mask is applied to the cost computation and back-propagation, and hidden states are not forward transformed into the next sensor frame.

With no egomotion information, one might expect the F1 measure for the baseline to be very poor. Surprisingly however, it does very well as illustrated in Figure 5. We suggest two explanations for this. Firstly, we posit that this is due to the dataset being relatively benign in terms of vehicle motion patterns. As most of the driving occurs down straight roads and at relatively constant velocity ($\sim$ 20 mph) the baseline may have learned a constant velocity model that could be used to correct the hidden state update. Our preliminary work on synthetic datasets reinforces this argument: training a model on a constant velocity vehicle achieves very good tracking performance, randomised sensor motion yields very poor performance. Secondly, the F1 measure here may be less informative, as the dataset is dominated by static objects such as walls and buildings. As such, a large fraction of the F1 score can be attained by learning the vehicle's motion and merely capturing static scenes. Nonetheless, the STM offers a clear improvement over the baseline DT in all future frames.

1) Qualitative Results: To qualitatively evaluate the model, we show a selection of sequences where the model does well, and where it does more poorly. As the dataset does not contain as many occlusions and for as long as the static set due to the setting of a vehicle driving through an urban environment, we look at the ability of the network to predict what happens in occlusion by maintaining the blacking out of the input every 5 frames at test time.

Figure 6 shows a compelling example of STM accurately tracking both dynamic and static objects through occlusion. In particular, the model accurately predicts the position of two static objects of different sizes when future frames are blacked out, and is also able to maintain the tracks when both objects are fully occluded.

Figure 7 illustrates how STM accurately predicts the tracks of both a moving vehicle (red circle) and two occluded pedestrians (orange rectangle) whereas the baseline model fails. For the latter, the predicted track of the vehicle gradually shifts from the ground truth until complete failure in frame 5 (false negative area), and it fails to separate and track the pedestrians through occlusion by the vehicle.

VI. CONCLUSION

In this paper, we have proposed an approach to perform object tracking for a mobile robot travelling in crowded urban environments, building on the previously proposed *DeepTracking* framework ([1], [2]). Crucially, unlike classical techniques which employ a multi-stage pipeline, this approach is learned end-to-end with limited architectural choices. By employing a *Spatial Transformer* module, the model is able to exploit noisy estimates of visual egomotion as a proxy for true vehicle motion. Experimental results demonstrate that our method performs favourably to *Deep-Tracking* in terms of accurately predicting future states, and show that the model can capture the location and motion of cars, pedestrians, cyclists, and buses, even when in complete occlusion.

Future work will look to estimate ego-motion, explore modalities such as radar, and extend the approach to 3D.

REFERENCES

[1] P. Ondrúška and I. Posner, "Deep tracking: Seeing beyond seeing using recurrent neural networks," in *The Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*, Phoenix, Arizona USA, February 2016.

[2] P. Ondrúška, J. Dequaire, D. Z. Wang, and I. Posner, "End-to-end tracking and semantic segmentationusing recurrent neural networks," *arXiv preprint arXiv:1604.05091*, 2016.

[3] T.-D. Vu, O. Aycard, and N. Appenrodt, "Online Localization and Mapping with Moving Object Tracking in Dynamic Outdoor Environments," in *Intelligent Vehicles Symposium, 2007 IEEE*, June 2007, pp. 190–195.
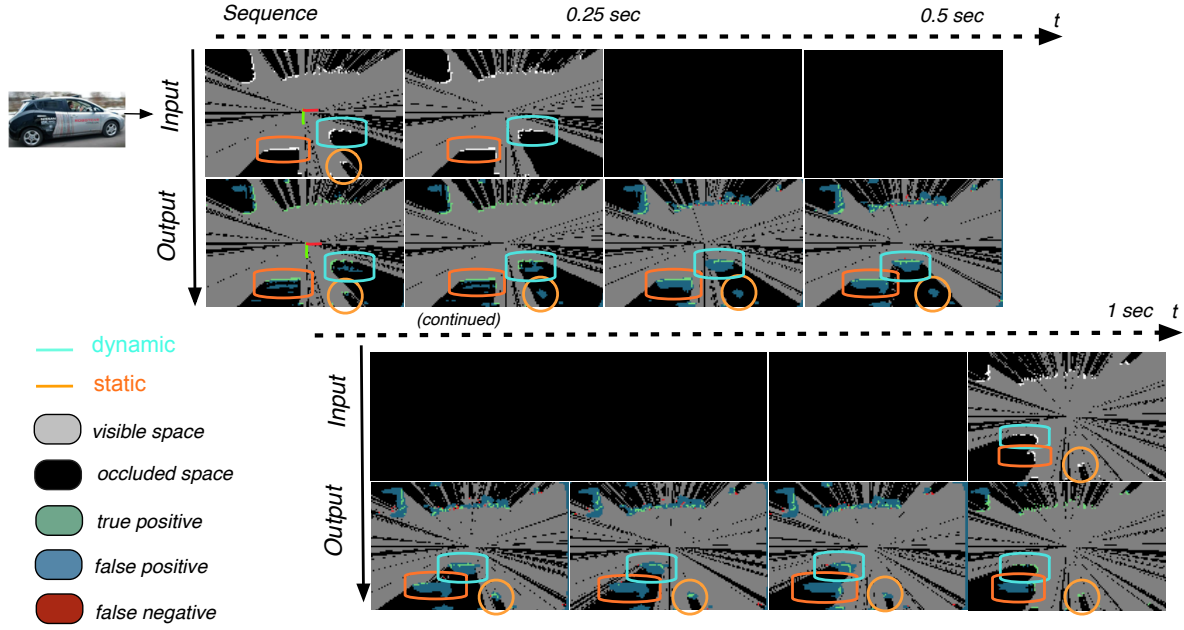
Fig. 6. Left to Right, Top to Bottom: Example sequence (1 second) of dynamic and static object tracking through occlusion by the STM. The output of the network is evaluated against the visible ground truth. We highlight several objects of interest that are correctly tracked through complete occlusion.
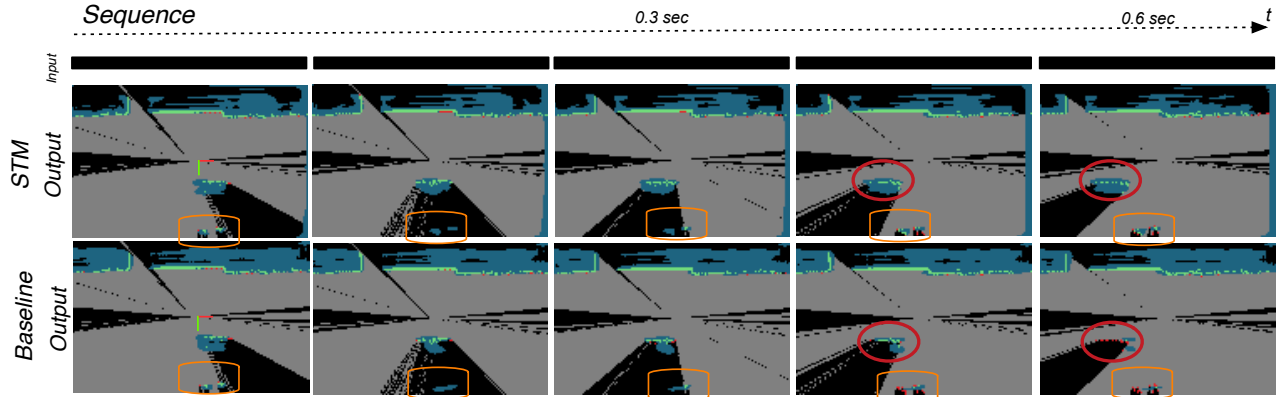

Fig. 7. Left to Right: Output sequence (0.6 seconds) of dynamic and static object tracking through occlusion as predicted by the Spatial Transformer Model (Top) and the Baseline Model (Bottom). We show the colour-coded comparison to the visible ground truth, with false negatives in red, false positives in blue, and true positives in green. Where the Spatial Transformer network consistently accurately tracks two pedestrians (orange rectangle) occluded by a moving vehicle (red), the Baseline model fails to capture the dynamics of both the vehicle and the pedestrians. This is particularly visible on the last frame, with most of the area around the actual location of these objects showing false negative occupancy prediction for the Baseline model.

[4] D. Z. Wang, I. Posner, and P. Newman, "Model-free detection and tracking of dynamic objects with 2d lidar," *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 1039–1063, 2015.

[5] A. Petrovskaya and S. Thrun, "Model based vehicle detection and tracking for autonomous urban driving," *Autonomous Robots*, vol. 26, no. 2, pp. 123–139, 2009.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[7] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 1, pp. 30–42, 2012.

[8] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng, "End-to-end text recognition with convolutional neural networks," in *Pattern Recognition (ICPR), 2012 21st International Conference on*. IEEE, 2012, pp. 3304–3308.

[9] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[10] S. Choi, K. Lee, and S. Oh, "Robust modeling and prediction in dynamic environments using recurrent flow networks."

[11] V. Patraucean, A. Handa, and R. Cipolla, "Spatio-temporal video autoencoder with differentiable memory," *arXiv preprint arXiv:1511.06309*, 2015.

[12] W. Lotter, G. Kreiman, and D. Cox, "Deep predictive coding networks for video prediction and unsupervised learning," *arXiv preprint arXiv:1605.08104*, 2016.

[13] M. Jaderberg, K. Simonyan, A. Zisserman, *et al.*, "Spatial transformer networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2017–2025.

[14] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[15] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. WOO, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in Neural Information Processing Systems*, 2015, pp. 802–810.

[16] M. Oquab, "Module for spatial transformer networks," https://github.com/qassemoquab/stnbhwd/.