End-to-End Tracking and Semantic Segmentation Using Recurrent Neural Networks

Peter Ondrúška
Mobile Robotics Group
University of Oxford, UK
ondruska@robots.ox.ac.uk

Julie Dequaire
Mobile Robotics Group
University of Oxford, UK
julie@robots.ox.ac.uk

Dominic Zeng Wang
Mobile Robotics Group
University of Oxford, UK
dominic@robots.ox.ac.uk

Ingmar Posner
Mobile Robotics Group
University of Oxford, UK
ingmar@robots.ox.ac.uk

Abstract—In this work we present a novel end-to-end framework for tracking and classifying a robot’s surroundings in complex, dynamic and only partially observable real-world environments. The approach deploys a recurrent neural network to filter an input stream of raw laser measurements in order to directly infer object locations, along with their identity in both visible and occluded areas. To achieve this we first train the network using unsupervised Deep Tracking, a recently proposed theoretical framework for end-to-end space occupancy prediction. We show that by learning to track on a large amount of unsupervised data, the network creates a rich internal representation of its environment which we in turn exploit through the principle of inductive transfer of knowledge to perform the task of it’s semantic classification. As a result, we show that only a small amount of labelled data suffices to steer the network towards mastering this additional task. Furthermore we propose a novel recurrent neural network architecture specifically tailored to tracking and semantic classification in real-world robotics applications. We demonstrate the tracking and classification performance of the method on real-world data collected at a busy road junction. Our evaluation shows that the proposed end-to-end framework compares favourably to a state-of-the-art, model-free tracking solution and that it outperforms a conventional one-shot training scheme for semantic classification.

I. INTRODUCTION

Complete and accurate situational awareness in complex, dynamic environments is a pivotal requirement for successful and safe robot operation. Often, however, this remains an elusive goal due to the limited field of view of the robot’s on-board sensors and to the complex and usually wide-ranging occlusions encountered. This limitation can impose significant challenges on the planner and may lead to otherwise unnecessarily conservative robot behaviour [17].

Object detection and tracking modules specifically addressing this problem are ubiquitous in robotics. Commonly, however, they feature multiple individual data-processing steps designed and optimised separately from one another. Traditional model-free approaches [21, 25, 22] make few assumptions with regards to the objects involved, such as their shapes or semantic characteristics, but they are often not robust. Model-based approaches [11, 15, 28] on the other hand limit the generality of these frameworks and often require separate object segmentation and classification steps.

In this paper we address this problem by introducing a novel, end-to-end trainable approach providing concurrent object tracking and recognition. It takes as input a stream of raw sensor scene observations that is often incomplete due to occlusions, and continuously provides estimates of the uncovered, occlusion-free scene containing information about the positions of all of the objects along with their classes as illustrated in Figure 1.

In doing so, we are inspired by the recently proposed framework of Deep Tracking [11] which leverages neural networks for end-to-end tracking. However, this framework has as yet only been deployed on predicting occupancy of comparatively benign, simulated data and using simple networks. We improve on this work in two ways. First, we motivate significant changes to the original Deep Tracking architecture [11] and demonstrate that these lead to substantial performance gains on complex, real-world scenarios. In particular, we propose to use multi-scale convolution to address the need to simultaneously track objects of different sizes, dynamic memory to effectively remember information for long periods of time, and static memory to learn place-specific information. Secondly, to effectively learn object semantics, we extend the framework

Fig. 1. Typical output of the proposed system capturing the situation around the robot in the form of a semantic map. The stream of raw sensor data is filtered by a recurrent neural network and produces classification of both directly visible and occluded space into one of several semantic classes.
II. RELATED WORKS

In this work we address the problem of effectively tracking the state of the environment around the robot. Classical approaches to this problem such as [9, 22, 26] typically involve a multi-stage tracking pipeline as illustrated in the top row of Figure 2. This pipeline features a sequence of explicit and largely hand-engineered steps consisting of object detection considering a stream of sensor input, semantic classification, data association, state estimation (including motion modelling) and, finally, occupancy grid generation. Instead, here we build on the recently proposed approach of Deep Tracking [11] featuring a recurrent neural network which directly maps from raw laser data to a semantically annotated and unoccluded occupancy grid. This is illustrated in the bottom row of Figure 2.

Deep learning approaches have been successful in a number of domains (see, for example, [6, 4, 23]) where they have benefited from large amounts of data in order to learn appropriate internal representations leading to significant performance gains above and beyond that achievable by classical methods. In our case the neural network is trained end-to-end to predict space occupancy and semantic labels directly from the raw laser data. While doing so it learns to perform an implicit tracking where the optimal internal representations about the hypotheses of moving objects and respective update procedures of classical tracking are inferred directly from the data.

To successfully apply deep learning, an appropriate neural network architecture for the task must be chosen. Abundant literature exists on the topic of finding optimal architectures for different tasks such as convolutional networks for image processing [19] and recurrent neural nets such as long short term memory [5] or gated recurrent units [3] for processing sequences. We propose a novel neural network architecture specifically tailored to real-world object tracking. The network...
shares similarity with architectures for semantic labelling of natural images [27] in terms of the ability to produce output of the same resolution. In addition we provide effective mechanisms to track objects of different sizes over time, learn place-specific information and recurrent mechanisms to remember information for long periods of time in order to track objects effectively even through long occlusions.

A common drawback of deep learning approaches is the need for large amounts of supervised data for training. We show that our network can be, in fact, trained very efficiently. The network first learns to track by just observing a stream of raw unlabelled sensor data and by trying to predict the next input. In turn we exploit the fact that the learned representation captures latent higher-order information in the data such that we can easily infer semantic labels for the tracked objects, using only a small amount of labelled data. This is a form of inductive transfer of knowledge between machine learning tasks [12]. In the context of neural networks it was successfully applied to a range of tasks, in the areas of multi-task learning [10, 2] and in the form of unsupervised pre-training [14, 7].

III. Problem Formulation

The input to our problem is a sequence of partially observed states of the world, computable directly from raw sensor measurements. We represent this state as a discretised 2D grid of size $M \times M$, parallel to the ground, built locally around the sensor. The partially observed state of the world is represented by two $M \times M$ binary matrices, collectively referred to as $x_t \in \{0, 1\}^{2M \times M}$. The first matrix encodes whether a cell is directly observable (value of 1, 0 otherwise) by the sensor at time $t$, while the second matrix encodes whether a cell is observed to be free (a value of 0), or occupied (a value of 1). The output we wish to obtain consists of two parts. The first part is an occlusion-free state of the world $y_t \in \{0, 1\}^{M \times M}$, represented by an occupancy map similar to the occupancy matrix in $x_t$. The second part is a semantic map $c_t \in \{1, ..., K\}^{M \times M}$, revealing, for each cell, which of $K$ types of objects (such as pedestrian, bicyclist, car etc.) is currently occupying it.

The problem therefore resolves to solving for $P(y_{t}, c_{t}|x_{1:t})$, the probability of the complete state of the world and its semantics at time $t$, given the observed input at all previous time steps $x_{1:t}$. This formulation can also be used to predict a future state $P(y_{t+n}y_{t+n}|x_{1:t})$ simply by providing an empty input for $x_{t+1:t+n} = \emptyset$.

In the next section we first outline a solution to the partial problem of estimating $P(y_{1}|x_{1:t})$ as suggested by the recently proposed Deep Tracking framework [11] but modified for operation in complex real-world scenarios. Then in Section IV we extend our solution to estimate $c_t$ via the application of the principle of inductive transfer.

1 Whereas $c_t$ is modelled for all the cells in practice, it is ignored for cells that are not occupied i.e. $y_{t}^{c} = 0$.

IV. Deep Tracking for Robotics

In this section, we focus on solving the first part of the problem formulated in Section III, namely uncovering the full, unoccluded state $y_{t}$ of the environment from the sequence of partially observed states $x_{1:t}$. We use the recently proposed Deep Tracking framework [11] to solve this problem. However, so far, this framework has been demonstrated only on simulated scenarios composed of simple geometric objects using a simple network architecture. A number of improvements to this architecture are needed to scale up its capacity to deal with complex, dynamic, real-world data encountered in robotics applications. We first briefly review details of the framework relevant to our application, then present our proposed improved architecture.

A. A Brief Review of Deep Tracking

Deep Tracking [11] is a method to model $P(y_{t}|x_{1:t})$ using a recurrent neural network [8]. Motivated by Hidden Markov Models [16], at a time $t$ a latent state $h_t$ is assumed to capture the complete information necessary for predicting $y_t$ (e.g. scene appearance and dynamics, locations of all objects, their shapes, velocities etc.), thus we have

$$P(y_{t}|x_{1:t}) = P(y_{t}|h_{t}).$$

Evolution of this latent state, which includes propagating model dynamics and integrating new sensor measurements, is modelled by an update operation

$$h_{t} = f(h_{t-1}, x_t).$$

The key element is that both the latent state update $f(h_{t-1}, x_t)$ and its decoding to the output $P(y|h_t)$ are modelled as parts of a single neural network and trained jointly. Equation 2 is modelled by the forward-propagation of information through hidden layers of the network, and Equation 1 is modelled by the decoding (output) layer (cf. Figure 4). Equations 1 and 2 can then be performed repeatedly as a form of recurrent neural network to continuously update the hidden state $h_t$ serving as a network memory and predict $y_t$, making it suitable for online stream filtering of sensor input.

When the ground-truth output $y_t$ is not easily available, as in our case, the network can be trained in an unsupervised fashion. Here, instead of optimising directly $P(y_t|x_{1:t})$, the network is trained to predict $P(y_{t+n}|x_{1:t})$ where $y_{t+n}$ is the part of $y_{t+n}$ that is directly observed in $x_{t+n}$. This is done by predicting $P(y_{t+n}|x_{1:t})$ and back-propagating [18] the error only on the observed part of the scene (cf. Figure 5). In other words, we train the network to correctly predict the subset of the ground-truth occupancy present in the future input. As demonstrated in [11] and also shown in Figure 8, an important consequence of this training strategy is that, at deployment, the trained network starts to correctly imagine objects and their movement in the occluded regions. This is because the situation with occluded input at deployment is similar to that at training when no input was provided at all for the future time $t+n$, and the network was trained to predict the observable regions.
V. SEMANTIC CLASSIFICATION THROUGH INDUCTIVE TRANSFER

In this section, we extend our solution to the partial problem $P(y_t|x_{1:t})$ presented in Section [IV] to the full problem of simultaneously estimating both occlusion-free occupancy and scene semantics $y_t$, $c_t$. We show that this can be achieved relatively easily by exploiting the knowledge the network has already learned to predict $y_t$, through the principle of inductive transfer [12]. The significance of this is that only a small amount of labelled training data is needed to allow the same network to master this additional task.

The clue resides in the hidden representation $h_t$ learned in the unsupervised training for tracking, which can be viewed as a universal descriptor of the state of the world. It captures not only the positions of individual objects, but also their motion patterns, shapes and other properties necessary for the successful prediction of scene dynamics. Because the network was trained to perform well in this task a reasonable assumption to make is that any information necessary for the prediction of the position of the objects in the near future must be already contained in this hidden representation. Object semantic class falls in this category as different objects differ mainly in their shape and motion patterns. Similar to predicting $y_t$ from $h_t$ in Equation [1], extracting $c_t$ can be achieved simply by building a classifier to predict $P(c_t|h_t)$.

### A. Training

Training the classifier to extract semantic information from $h_t$ is not straightforward as a supervisory signal would need to be provided for all the pixels whether they contain an actual object or not and for both visible and occluded areas. Such a supervised dataset would be very difficult to produce if only occluded raw laser scans are available.

Instead we label only the visible cells of the available raw input data which contain an actual obstacle. Then we predict labels for all the pixels but back-propagate the error only on those with a label. This principle makes intuitive sense, however could result in the classifier to rely too much on the part of memory $h_t$ affected by presence of visible input $x_t$ and performing well only for directly visible parts of the scene but poorly for the occluded objects where the prediction is driven purely by the previously remembered information in $h_t$. To address this issue a more elaborate training procedure is necessary.

To ensure good performance of the network on classifying occluded objects it must be trained in such settings. This can be achieved using the same principle used to train the network to predict $y_t$ from $h_t$. We train the network to predict the future semantic label $c_{t+n}$ while providing only input $x_{1:t}$ which forces the network to use information stored in memory $h_t$ and then back-propagate the error compared to the true label of $x_{t+n}$. The entire process is illustrated in Figure [IV].

VI. THE NETWORK ARCHITECTURE

The simple recurrent neural network proposed in [III] was demonstrated to be sufficient for the simulated dynamic scenario evaluated in that work. However, an effective deployment in real-world robotics applications poses a set of challenges for which a more appropriate architecture must be chosen. In particular it requires the ability to simultaneously track objects of different sizes such as cars and pedestrians, to effectively remember information for long periods of time to deal with occlusions, learn and exploit place-dependent information such as the presence of static obstacles and lastly produce output for both space occupancy and it’s label. We therefore in this section present a new network architecture designed to address the above issues.
An overview of our proposed network is depicted in Figure 4. The input $x_t$ at time $t$ is processed by a multi-layer network. At each layer the output of the previous layer is combined with its own activations at time $t - 1$ implementing the recurrence. This allows the network to extract and remember the information from the past and use it for prediction at time $t$. The output of the final layer is then converted into the resulting output $y_t$ through simple convolutional decoder.

Unlike classical convolutional networks such as [20] this network does not feature max pooling and maintains the same resolution in each layer. In addition it features four key elements critical for successful tracking and classification in realistic scenarios: multi-scale convolution, dynamic memory, static memory, and pair of decoders which we describe below.

1) Multi-Scale Convolution: For the network to correctly predict the occupancy and label at location $i$, such as affected by the presence of a moving object, this object must fall in the receptive field of the neuron in the final layer. The receptive field is the part of the input affecting the value of the neuron. In the case of classical convolution the receptive field is the $K \times K$ neighbourhood where $K$ is the size of the convolution kernel. The size of the receptive field however limits the size of effectively tractable objects in the input which can be of vastly different sizes in realistic settings. One way to increase the receptive field is to increase the kernel size or stack multiple convolutions on top of one another. This however creates a computational challenge as the number of parameters increases exponentially with the number of layers. This gives a $(2K+1-1) \times (2K+1-1)$ receptive field at final layer $K$. This dilated convolution is then used as an elementary computation step to implement the dynamic memory described below.

Instead we use a stack of dilated convolutions [27] where the receptive field grows exponentially with the number of layers. The basic idea is to perform the classical $3 \times 3$ convolution but skipping $2^{k-1} - 1$ pixels in between convolved pixels at layer $k$, as illustrated in Figure 5. This gives a $(2^k+1-1) \times (2^k+1-1)$ receptive field where $K$ is the size of the convolution kernel. The size of the receptive field however limits the size of effectively tractable objects in the input which can be of vastly different sizes in realistic settings. One way to increase the receptive field is to increase the kernel size or stack multiple convolutions on top of one another. This however creates a computational challenge as the number of parameters increases exponentially with the number of layers. The receptive field grows exponentially with the number of layers.

2) Dynamic Memory: To be able to track a moving object through extended periods of occlusion, the network must remember the location of the object and other properties such as its shape and velocity. Findings from studies on recurrent neural networks stress out the importance of specially dedicated units such as long short term memory [5] to support information-caching, as otherwise training suffers from the vanishing gradient problem [13]. Inspired by [24], we implement a convolutional variant of gated recurrent units [3] as the processing step at each layer. The output of each unit is given by the weighted combination of its previous output at time $t - 1$ and a candidate memory $\tilde{h}_t$ computed from the output of the layer below with the forgetting of information controlled by the reset gate $r_t$:

$$f_t = \sigma(W_{xz} \ast x_t + W_{hz} \ast h_{t-1} + b_x),$$  \hspace{1cm} (3)

$$r_t = \sigma(W_{xf} \ast x_t + W_{hr} \ast h_{t-1} + b_r),$$  \hspace{1cm} (4)

$$\tilde{h}_t = \tanh(W_{xh} \ast x_t + r_t \circ W_{hh} \ast h_{t-1} + b_h),$$  \hspace{1cm} (5)

$$h_t = f_t \circ h_{t-1} + (1 - f_t) \circ \tilde{h}_t.$$  \hspace{1cm} (6)

Here $\ast$ denotes dilated convolution described earlier and $\circ$ denotes element-wise multiplication.

3) Static Memory: We allow each cell to learn a unique and universally accessible piece of information different from all other cells. This is achieved by biases $b_z$, $b_r$, and $b_h$ in Equations 3-5 which are not a per-layer constant as in the case of classical convolution, but are learned individually for each neuron during the training. As shown in Section VII this allows the network to learn place-specific information such as the static occupancy of the cell or the usual motion patterns and classes in a particular area, which can then be used to aid the network prediction.

4) Decoders: Finally, we employ pair of simple convolutional decoders to decode output of the final layer to the cell occupancy $y_t$ and class label $c_t$. The difference is that for the class label we employ softmax (multinomial logistic
regression) output for the $K$ classes instead of a sigmoid function.

VII. RESULTS

In this section we demonstrate the efficacy of the proposed system in both tasks of tracking and semantic classification in a complex, real-world scenario. We show that the trained network is able to track and classify a variety of different objects even through complete occlusion, and is able to predict the evolution of the scene in the near future. In particular, we are interested in the relative performance of the proposed network architecture in correctly predicting both object positions and class labels. We show that the achieved tracking accuracy is superior to the original architecture presented in [11], as well as that of an alternative state-of-the-art model-free tracking solution targeted at the same problem [22].

A. Dataset

We collected a 75 minute long log from a stationary robotic platform equipped with a Hokuyo UTM-30LX 2D laser scanner, positioned in the middle of a busy urban intersection, as depicted in Figure 6. 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 the complete scene was fully observable. We subsampled the dataset at 8Hz and split it into a 65 minute unsupervised set to train the network, and a 10 minute long test set to measure the occupancy prediction performance. In addition, we hand-labelled 800 scans from the training set into 4 classes for the purpose of network semantic training, and 200 scans from the test set for the evaluation of its semantic classification accuracy. The classes considered are: pedestrian, car/bus, cyclist and background (static obstacle).

The input to the network $x_t$ (the partially observed occupancy grid) is computed from raw 2D laser scans by ray-tracing. Cells where a laser 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. We build a $100 \times 100$ grid around the sensor, and each cell covers a $20 \times 20$ cm$^2$ area, the input grid thus spanning a total area of $20 \times 20$ m$^2$.

B. Network training

We tested the proposed network architecture described in Section VI configured with three hidden layers of 16 channels per layer. This network has a total number of 1,010,193 parameters and was trained for 72 hours until convergence on a single Nvidia Titan GPU with 6 GB of memory, using the unsupervised training procedure described in Section IV-A. The training sequence was split into mini-batches of length 40 (5 seconds of stream). For every mini-batch, the network is shown 10 frames and trained to predict the next 10 frames, leading to two such sequences per mini-batch. This was chosen to cover the usual length of the occlusions in the scene and we expect it would need to be increased for longer-lasting occlusions.

Next, the classifier of $P(c_t|h_t)$ was trained in just 2 hours on the semantic classification task using the 2 minute long labelled dataset. Because the dataset is skewed and contains

![overhead image of the robot environment](image_url)

![predicted obstacle probability](image_url)

Fig. 7. Trained network output when provided no input [bottom] and corresponding aerial view of the robot environment [top]. The ability of the network to learn per-pixel information allows adaptation to the training environment. This allows the network to confidently predict the position of static obstacles such as buildings, as well as the probability of any given cell being occupied even without any sensor input. Pavements show higher probabilities than the centre of the roads. For clarity of visualisation, we show here the log of the probabilities of occupation.
many more objects of a particular type, e.g. pedestrians, we used a weighting scheme assigning class weights equal to the inverse of class frequency.

C. Benchmarking Against an Existing Approach

To compare the performance of the proposed end-to-end system to more traditional multi-stage pipelines, we evaluate the ability to predict future movement of dynamic objects of the proposed framework against a recently proposed state-of-the-art approach [22] based on model-free tracking of dynamic objects using a Kalman filter. This method accepts raw laser scans, performs data clustering and association, as well as velocity estimation of moving objects. This information is then used to predict the positions of individual points in the future. We tuned the parameters of this method on the training set, then collected results from the test set, and converted the output into occupancy grids for comparison with the proposed method.

D. Evaluation

Two typical input sequences and their corresponding predicted network output are shown in Figure 8. The network is able to uncover the unoccluded scene including the space occupancy $y_t$ and object labels $c_t$. Moreover it is able to update the positions of dynamic objects through temporary occlusion demonstrating that it has learned to track and recognise objects in the scene.

In what follows, we quantitatively evaluate the performance
of the proposed system to justify these qualitative observations.

1) Occupancy Accuracy: As the ground-truth occupancy annotations $y_t$ of the full unoccluded state was not available, we, instead, measured the accuracy of predicting the future occupancy $y_{t+n}$ with respect to the visible part of the scene in $x_{t+n}$. This is the same metric as the one used to train the network in Section IV. For binary obstacle prediction, we show in Figure 2 the computed F1-scores when predicting 10 consecutive frames, averaged over the test set, and compare to both the original architecture described in [11] (that we train in a similar manner), and to the state-of-the-art multi-stage pipeline approach of [22]. The prediction is accurate in the near horizon and progressively decreases over time. This is expected, as the uncertainty of the state of the world increases with the prediction horizon. In both cases our results outperform the two alternative approaches, demonstrating the effectiveness of the proposed network architecture and advantages of end-to-end learning.

Another experiment is concerned with measuring the effectiveness of the desired ability to learn place-specific information. One way to evaluate this is to visualise the network prediction $y_t$ without providing it with any input as displayed in Figure 7. Even without input sensor information, the network is able to provide an estimate of the expected occupancy probabilities, which is higher at the locations of static obstacles and at crowded areas of the scene such as pavements. As no propagation of the information through the network occur this is clearly only made possible by the ability of the network to remember this information in its static memory during training.

2) Semantic Accuracy: To quantify the network’s ability to classify scene semantics, we compute the confusion matrix which is shown in Figure 10. As can be seen, the network is able to produce reliable classification for the object classes considered. The main source of error lies in distinguishing cyclists from pedestrians, as they often exhibit similar shapes in 2D laser data.

To verify the value of the proposed inductive transfer of knowledge, we compare this result to an alternative approach of classifying scene semantics, which takes the form of a one-shot classification of all directly visible obstacles from a single raw-sensor input. As a representative solution to such an approach, we trained a three-layer deep convolutional classifier predicting $c_t$ directly from $x_t$. Despite conducting rigorous parameter tuning, classifying directly the input $x_t$ yields inferior classification accuracy compared to classifying the hidden representation $h_t$, the negative log likelihood of correct labels being respectively $\sum - \log P(c_t | x_t) = 101.967$ and $\sum - \log P(c_t | h_t) = 49.129$. This demonstrates that $h_t$ offers a powerful semantic descriptor of the scene and can be used as input for accurate semantic classification.

3) Timing: The forward propagation of a single input through the network takes 15ms on an Nvidia Titan GPU and 83ms on a commodity laptop CPU. This is sufficient to enable real-time processing of the considered stream of laser data at 8Hz.

VIII. CONCLUSIONS

In this work, we presented a novel end-to-end trainable solution for real-time object tracking and classification in complex and partially-observable real-world environments. Leveraging the representational power of recurrent neural networks and employing efficient training procedures, the method surpasses a representative state-of-the-art model-free method, while substantially reducing the requirement for hand-engineered knowledge.

The method can be extended or applied in a number of ways. The universal schema of input and output opens up a possibility to apply the method to situations beyond those evaluated, such as accounting for robot motion which can be handled by moving the robot inside the grid, or multi-sensor or multi-robot fusion. Additionally, the knowledge of the environment captured in learning to track can be further exploited to provide different kinds of semantic information such as bounding boxes. Finally, the inherent ability to predict the future evolution of the environment around the robot can be leveraged upon in more far-sighted planning scenarios.