Forecasting Battery State of Charge for Robot Missions

Ameer Hamza and Nora Ayanian
Department of Computer Science
University of Southern California
Los Angeles, CA, USA
{ahamza, ayanian}@usc.edu

ABSTRACT

Due to limited power onboard, a significant factor for success of distributed teams of robots is energy-awareness. The ability to predict when power will be depleted beyond a certain point is necessary for recharging or returning to a base station. This paper presents a framework for forecasting state of charge (SOC) of a robot's battery for a given mission. A generalized and customizable mission description is formulated as a sequence of parametrized tasks defined for the robot; the missions are then mapped to expected change in SOC by training neural networks on experimental data. We present results from experiments on the Turtlebot 2 to establish the efficacy of this framework. The performance of the proposed framework is demonstrated for three distinct mission representations and compared to an existing method in the literature. Finally, we discuss the strengths and weaknesses of feedforward and recurrent neural network models in the context of this work.

CCS Concepts

•**Computer systems organization → Robotics;** *Sensor networks;*
•**Hardware → Power estimation and optimization;** Batteries;

Keywords

Robotics; Mobile robotics; Energy awareness; Persistent robotics; Battery modeling

1. INTRODUCTION

Uninterrupted, reliable and persistent operation is a significant consideration for distributed teams of robots, as well as individual robots working cooperatively with others. For distributed teams of robots, communication is often limited due to distance or cost, and running out of energy while in the field can cause severe damage or loss of the asset. For cooperative robots, depleting on-board energy can result in total mission failure. Power management for battery-powered robots is especially important for persistent multi-robot systems [11, 20]. Finally, as the push for building ever smaller robots further limits usable power, it becomes critical to forecast the power response of robots in multi-robot missions.

Forecasting the mission-specific power consumption of a robot is necessary to complete missions reliably, smoothly, and efficiently. Such a forecast can be used to predict and avoid power related failures. Moreover, it may be helpful in *a priori* evaluation of mission design with respect to robot parameters, or determination of required on-board power for a particular mission. For robots operating in hazardous or inaccessible environments, predicting power consumption can prevent loss of expensive hardware. Besides improving reliability, it can also contribute to efficient decision making and planning, as well as more realistic and accurate power-consumption models for robots in simulation.

In this work, we consider the problem of forecasting the remaining battery capacity for a robot. State of charge (SOC) is the most common metric used for remaining battery capacity and is measured in percentages representing the complete battery cycle from 100% (fully charged) to 0% (fully discharged).

This paper introduces a framework for mission-specific energy-awareness in mobile robots, aimed at enabling robots to forecast energy consumption for a mission by learning from past missions. *The contribution of this work* is a generalized framework for forecasting mission power consumption in battery-powered robots. Existing work in the literature focuses on specific types of robots, e.g., underwater, whereas this framework can be applied to any type of battery-powered robot. We explore the efficacy of this framework for three different mission representations, encoding varying amounts of detail. We execute a number of missions on a battery-operated ground robot and collect the relevant data, including the measured SOC during mission execution. We use this data to train two types of neural networks (*recurrent* and *feedforward*) in order to map actions to the change in SOC caused by their execution. Finally, we discuss the ability of these models to forecast the SOC for the duration of a mission under different configurations, and compare to an existing method in the literature.

2. LITERATURE REVIEW

The need for intelligent battery monitoring and management has been extensively discussed, especially in the context of electric automobile applications [12, 21]. Determining the remaining capacity of a battery has been a major objective for a number of electric circuit-based battery modeling methods [1, 3, 5, 7–9, 18]. Equivalent electrical circuits are designed to model batteries using different electrical components [7], including resistors for internal resis-

SAC'17, April 3-7, 2017, Marrakesh, Morocco
Copyright 2017 ACM 978-1-4503-4486-9/17/04…$15.00
http://dx.doi.org/xx.xxxx/xxxxxxx.xxxxxxx

tance [8], suitable RC circuits for dynamic loads [3,9], diodes for switching between equivalent circuits for charging and discharging [18], and Zener diodes to model battery response for a range of load currents [1]. A number of data-driven models for reporting remaining battery capacity, offline and on-line, have been explored [2,6,15,16,19,22]. Some methods employ neural networks for approximating battery models using voltage, load currents, internal resistance and temperature inputs [19,22]. Others also include remaining capacity as input to improve predictions [15]. A trained network can provide *a priori* estimates for a Kalman filter for on-line SOC estimation [2]. Both the circuit-based and data-driven modeling methods described can be used to predict battery state of charge given precise load characteristics (current, voltage, temperature, etc.). However, for robots it is often infeasible to estimate these load characteristics in advance for a mission.

Various approaches characterize robots' power consumption in relation to the environment. Pentzer *et al.* characterize terrain by estimating parameters for skid-steer robots [14]; this requires knowledge of vehicle geometry and mass distribution. Sadrpour *et al.* apply a longitudinal dynamics-based linear regression model and a Bayesian regression model, both based on prior terrain and driving style information, for ground vehicles [17]. LeSage *et al.* cluster recorded loads for different terrains in order to characterize a range of power demands and using battery models to predict battery power requirements [10] . These methods rely on knowledge of the physical parameters of the robot (e.g., vehicle dynamics, weight distribution) and environment.

In the present work, we do not rely on knowledge of physical parameters; instead, we investigate ways to use common mission representations to predict battery response. If physical parameters are available, they can be used to train the model, and thus improve the prediction. If environmental information is not available, our method allows tuning previously learned models on a small amount of mission execution data (e.g., loss of 10% SOC) to significantly improve predictions. This makes our framework readily implementable without precise knowledge of the robots' physical properties and operating environment.

Other methods measure power consumption of individual components of a robot in order to estimate parameters for a power consumption model. One approach estimates linear model parameters for power estimation of an underwater robot, by executing a sequence of trajectories influencing only one degree of freedom [4]. Parasuraman *et al.* build a model by measuring power consumption while running a sequence of single-component operations, then estimate the total power use as the sum of the power consumed by all of the components [13]. These methods depend on prior knowledge of how various components of the robot operate during execution of a given mission, which may require detailed knowledge of the functioning of the robot. We present a comparison of our framework with that of Parasuraman *et al.* [13] in Section 5.5.

Using environment-based and component-based models requires the robot's assigned mission to be broken down into terrain-specific or component-wise operations, respectively, which may be infeasible. In this work, we consider a robot and its operating environment as a whole, and model power consumption as a function of the mission representation. We propose a generalized framework for forecasting power requirements by using information typically available in mission descriptions. This makes our method more readily applicable across platforms and environments since characterizing loads or estimating the physical parameters of the robot and operating environment are not necessary.

3. PROBLEM FORMULATION

Let a job-set J be a set of atomic operations (e.g., turn on motor, transmit data, etc.) defined for a robot r such that any tasks allocated to the robot can be completely described as combinations of jobs in the set. Each job element $j \in J$ is defined in terms of parameters $param(j)$. We define a mission of length T as an ordered sequence of actions $M = \langle a_1, a_2, \ldots, a_T \rangle$, where an action a_i is an ordered vector containing the parameters for all the jobs in the robot's job-set $a_i = \langle param(j_1), param(j_2), \ldots, param(j_n) \rangle$.

To illustrate, consider a robot capable of moving forward a distance x at velocity v and idling for duration t. We can define a job set $J \equiv \{idle, move\}$, and the parameters $param(j_1) \equiv t$ and $param(j_2) \equiv (x, v)$, resulting in action $a_i = \langle t_i, x_i, v_i \rangle$. A mission of length $T = 2$ to move forward for 3 m at 1 m/s, then idle for 10 s would be represented with two actions $a_1 = \langle 0, 3, 1 \rangle$ and $a_2 = \langle 10, 0, 0 \rangle$, as $M = \langle a_1, a_2 \rangle$.

Defining a job-set is an engineering decision, depending on the type of robot, operating environment, and nature of the mission. The job-set J must consist of jobs that suitably encode parameters that significantly impact power consumption. For example, terrain-related parameters must be encoded in the mission description, in order to predict the SOC in multi-terrain scenarios. There are trade-offs in selecting different levels of abstraction for a job-set, which we describe in Section 3.3.

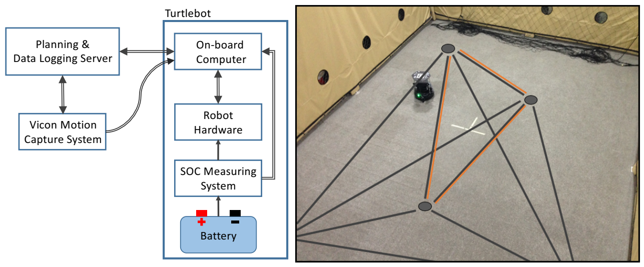
We use this formulation to build a model f that maps action a_i to the resulting change in SOC, $f : a_i \rightarrow \Delta SOC(i)$, where $\Delta SOC(i) = SOC(i) - SOC(i-1)$. Ideally, for a given mission M and the current action a_i, we compute the *predicted state of charge*, $SOC_p(i+k)$, of the robot's battery after k future actions in the mission, such that the maximum prediction error $\varepsilon \geq |SOC_p(i+k) - SOC_m(i+k)|$ is minimized, where $SOC_m(i+k)$ is the measured SOC. However, the stochasticity introduced by errors in actuation, wheel slippage, execution on a physical system, etc., makes providing such a bound impossible. Thus, we evaluate our model experimentally, by running a number of missions, then using networks trained on each of those missions to predict power consumption in other missions, and present representative results here. We also compare to one of the additive models in the literature, showing that our method provides better predictions.

We demonstrate the efficacy of this framework for missions described by three different job-set definitions of varying levels of abstraction. Twelve missions were executed and the following data were collected for each mission: action parameters, robot pose, battery statistics, time-stamps,ăand the measured SOC during mission execution. We use this data to train two types of neural networks to map actions to the change in SOC caused by their execution. We also explore the ability of these learned models to forecast the SOC for the duration of a mission. A detailed description of the methods used follows.

3.1 Experimental Setup

Missions were executed on a Turtlebot 2 with additional 6W geared DC motor and a 3W LED to provide additional modes of power consumption beyond locomotion. A netbook onboard the robot was used for control, measurements, and communication.

Figure 1a shows a schematic of the experimental setup. Two lithium-



(a) (b)

Figure 1: (a) Overview of experimental setup. (b) Robot operating area for missions described in the experiments. A 3-waypoint polygon is shown (orange) as an example of a High-level action.

ion batteries (4400 mAh, 14.8 V) with distinct states of health (one new, one old) were used to explore the adaptability of learned models to changes in battery health. To measure battery voltage, current, temperature, and SOC, a Texas Instruments BQ76930 battery monitor was installed between the battery and the robot hardware. The circuit module combines voltage mapping with Coulomb counting and temperature compensation to estimate SOC every 250 ms.

The operation area for mission execution (Fig. 1b) was equipped with a 12 camera VICON MX motion capture system to record data related to locomotion and provide pose information to the motion controllers and other software on the robot. An independent server published the mission as a sequence of actions and logged data.

3.2 Mission Description

Missions are encoded as a sequence of parametrized actions. All missions are based on the robot traversing a series of waypoints in the following fashion: the robot starts from rest at waypoint w_i, turns by angle θ with angular velocity ω towards the next waypoint w_{i+1}, then moves distance x at average speed v towards w_{i+1}, and comes to rest when w_{i+1} is reached. The paths are the edges of a fully connected graph defined on these waypoints (see Fig. 1b). In our missions, edges have a corresponding speed and parametrized $clean(pwm_c)$ and $beam(pwm_b)$ operations, which modulate the operating power for the onboard DC motor and high-powered LED, respectively. Each mission is a combination of jobs in a job-set.

3.3 Job-set Definitions

Forecast performance was studied with respect to three different mission representations, Low-level, Mid-level, and High-level.

The Mid-level Job-set (J_m), with five jobs, arises as a natural implementation of robot functions for execution of the type of missions we have described. $move(x, \Delta z, v)$ moves the robot forward distance x at velocity v and altitude difference between start and end points Δz (while our missions were run on fairly flat ground, it is not completely flat, thus we take height change into account). This trajectory is susceptible to error, thus it is governed by feedback controllers. $turn(\theta, \omega)$ turns the robot in place by $\theta \in [0, \pi]$ radians with angular velocity ω. $beam(pwm_b)$ modulates the power provided to the high-power LED according to $pwm_b \in [0, 100]$. $clean(pwm_c)$ modulates power to the DC motor according to $pwm_c \in [0, 100]$. $idle(t_d)$ keeps the robot idle for t_d seconds.

The High-level Job-set (J_h) encodes the robot's functionality more compactly, as one job: $traverse\_polygon(x^h, v^h, \theta^h, \omega^h, n)$ causes the robot to traverse a polygon (sub-graph) with n waypoints for total distance $x^h = \sum_{i=1}^{n} x_i$, weighted average speed $v^h = \frac{1}{x^h}\sum_{i=1}^{n} x_i v_i$, total turning angle $\theta_h = \sum_{i=1}^{n} \theta_i$, and weighted average angular speed $\omega_h = \frac{1}{\theta_h}\sum_{i=1}^{n} \theta_i \omega_i$ (a three-waypoint example is shown in Fig. 1b). Note that this job can be represented as a sequence of *turn* and *move* jobs from the Mid-level Job-set, but without the change in altitude from the *move* job. $beam\_polygon(pwm_b^h)$ defines the operating power of the LED while a polygon is being traversed, according to a weighted average power given by $pwm_b^h = \frac{\sum_{i=1}^{n} x_i pwm_b^i}{\sum_{j=1}^{n} pwm_b^j}$. $clean\_polygon(pwm_c^h)$ modulates power to the DC motor while the polygon is being traversed, governed by a weighted average power given by $pwm_c^h = \frac{\sum_{i=1}^{n_h} x_i pwm_c^i}{\sum_{j=1}^{n_h} pwm_c^j}$. Again, $idle(t_d)$ keeps the robot idle for t_d seconds. The High-level Job-set parameters statistically combine the mission parameters associated with the sequence of edges representing the polygon in the graph, gaining compactness at the cost of accuracy.

The Low-level Job-set (J_l) encodes the physical parameters of mission execution more precisely. $traverse\_segment(x^l, v^l, v^l + \Delta v^l)$ attempts to encode the actual trajectory followed by the robot instead of a straight line by segmenting an edge in time. Thus, a *move* job in the Mid-level Job-set is represented in the Low-level Job-set as a sequence of $traverse\_segment$ jobs where each job is parametrized in segment length x^l, initial velocity v^l, and final velocity $v^l + \Delta v^l$. For our experiments, these segments were based on the data (distance, velocity, SOC etc.) collected every 250 ms. For instance, if the robot takes 5 s to traverse an edge in the graph, 20 segments are created for the edge and the job parameters are derived from the collected data for these segments. Note that these parameters reflect the velocity profile which changes across the edge, resulting in a more precise encoding of the velocity in comparison to the average velocity used for equivalent mid-level jobs. $turn(\theta, \omega)$, $beam\_segment(pwm_b)$, $clean\_segment(pwm_c)$ and $idle(t_d)$ are defined in the same way as in the Mid-level Job-set, but on these smaller segments. Note that this job-set definition results in larger mission representations that explicitly encode the desired trajectories for the robot, which may not be feasible in many real-world deployments, due to the the difficulty in knowing the mission or terrain in detail, and the size of the mission definition. However, we include this representation since it encodes entire trajectories instead of just endpoints, so it automatically encodes deviations from straight line travel (due to actuation error, slight changes in heading, etc.), and leads to models that have learned on and can be applied to curved trajectories.

4. EXPERIMENTS

Twelve missions were designed for experimentation. Execution of each mission took between two to three hours and resulted in varying final battery SOC. Each mission was executed twice, with batteries of different states of health, resulting in 12 pairs of data-sets. The three job-set representations were generated for each mission. During execution, parametrized action vectors were recorded for all three representations along with battery statistics and timestamps.

To evaluate the framework's predictive capabilities, each experiment was conducted using a complete record of one mission as training data and data recorded from the remaining missions as testing data. Thus, the total number of training and testing sets for a single battery were 10 and 90, respectively. Two types of neural networks were used to learn models mapping action parameters

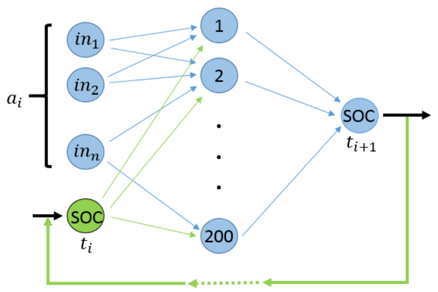


Figure 2: Internal structure of neural networks used with action parameters as inputs and the resulting SOC as output. Blue nodes and links represent the feedforward neural network. For recurrent networks, the input layer also includes a *context unit* (green), which encodes SOC predicted by the network after previous action a_{i-1}.

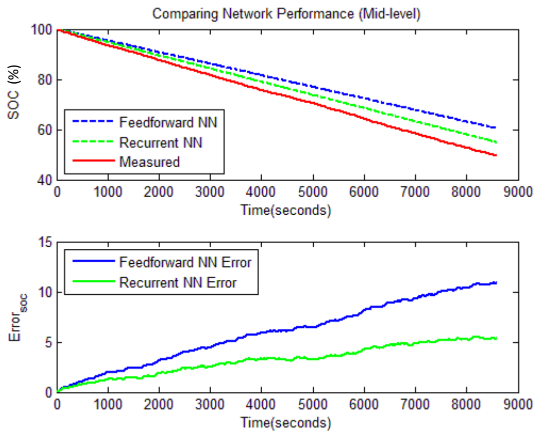


Figure 3: Forecast performance in recurrent and feedforward networks.

to resulting SOC: *feedforward networks* and *simple recurrent networks*. The number of input neurons in these networks varies depending upon the length of the action parameter vector. The number of neurons in the hidden layer was set to 200 whereas the output layer consisted of one neuron.

A feedforward network takes an action parameter vector a_i as input and predicts the resulting drop in the state of charge $\Delta SOC_p(i)$. The predicted state of charge $SOC_p(i+k)$ after execution of a sequence of actions $\langle a_i, a_{i+1}, ..., a_{i+k}\rangle$ is given by $SOC_p(i+k) = SOC_p(i-1) + \sum_{j=i}^{k} \Delta SOC_p(j)$, where $i = 1,\ldots,T$, $k = 0,\ldots,(T-i)$.

Blue nodes and links in Fig. 2 represent the feedforward neural network. Note that the predicted output is not fed back into the feedforward network. Recurrent neural networks can be created by adding a *context unit* (green) which enables the network to use its own prediction at the previous time-step as input. The complete network shown in Fig. 2 is the recurrent neural network. Thus, the recurrent network maps action parameters a_i and the previous state of charge $SOC(i-1)$ to the resulting state of charge, $f : (a_i, SOC(i-1)) \rightarrow SOC(i)$ such that $SOC(i) = f(a_i, SOC(i-1))$. A future state of charge forecast $SOC_p(i+k)$ for a sequence of action parameters $\langle a_i, a_{i+1}, \ldots, a_{i+k}\rangle$ is computed recursively: $SOC_p(t_{i+k}) = f^k(a_{i+k}, f^{k-1}(a_{i+k-1}, \ldots, f^1(a_i, SOC(i-1))\ldots))$,

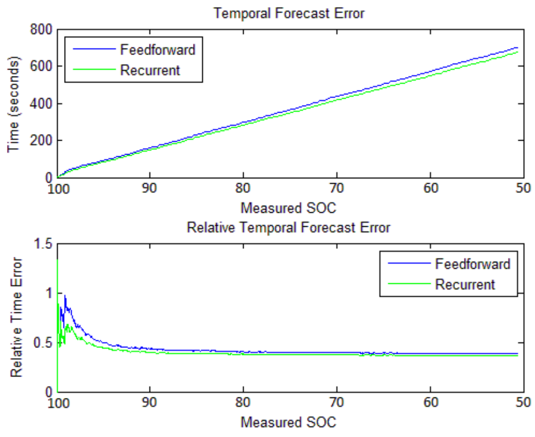


Figure 4: Absolute and relative temporal errors for both networks.

where $i = 1,\ldots,T$ and $k = 0,\ldots,(T-i)$. Henceforth, we use *experiment* to refer to the process of training a network on data recorded from a mission and using the trained network to forecast the SOC for another mission.

5. RESULTS AND DISCUSSION

5.1 Recurrent Neural Networks

A trained recurrent neural network takes the i^{th} action parameter along with the state of charge before the action is performed $SOC(i-1)$ as input, and outputs the predicted state of charge after the action has been performed $SOC_p(i)$. Fig. 3 shows the forecast performance of the recurrent network trained on mission dataset M5 and tested on M4, represented with the Mid-level Job-set, by superimposing the forecast SOC_p (green dashed) on the measured state of charge SOC_m (red solid). Absolute error $\varepsilon_a(i) = |SOC_m(i) - SOC_p(i)|$ is also shown in Fig. 3 (bottom). Note that this forecast is generated by feeding the network's SOC *prediction* back to its input for the next prediction.

We are also interested in estimating the time it will take the battery to reach a certain SOC during a mission. Figure 4 shows the absolute and relative temporal forecast errors given by $\varepsilon_t(SOC) = |t_m - t_p|$ and $\varepsilon_r(SOC) = \frac{|t_m - t_p|}{t_m}$, respectively. Here, t_m and t_p denote measured and predicted times for the battery to drop to a certain SOC, respectively. Figure 4 shows that the temporal errors accumulate as a mission progresses which may result in higher final temporal error for longer missions.

5.2 Feedforward Neural Networks

The input to a feedforward network is an action parameter vector a_i, and the output is the predicted change in state of charge ΔSOC. Figure 3 includes the forecast and errors of the feedforward networks (blue dashed) trained and tested on the same data-set to provide a quantitative comparison of the two networks' performance.

Recurrent neural networks outperform feedforward networks (i.e., they are more accurate) for all mission pairs, demonstrated in the box-plot of mean absolute errors in Fig. 5. This is because recurrent neural networks also take into account the previous SOC in addition to the action parameters for their forecasts.

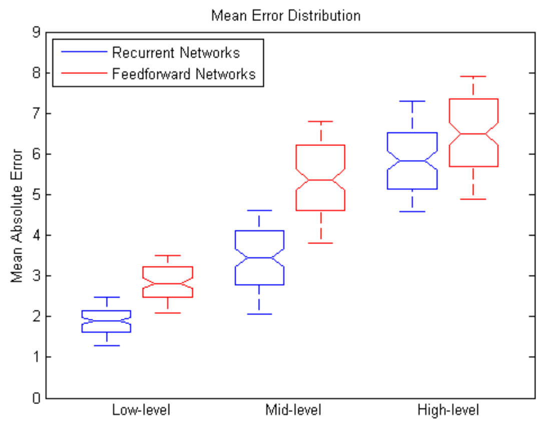


Figure 5: Box-plot comparison of mean absolute errors for three mission representations.

5.3 Adaptability

While recurrent networks perform better than feedforward networks, feedforward networks offer advantages due to their structure. Ideally, a previously learned network should quickly adapt to changes in the robot (e.g., a new battery) while executing a mission. Recall that each mission was executed with two different batteries to evaluate the adaptability of a trained network to changes in system parameters. We extract partial data for mission execution for a 10% drop in SOC and use it to tune a previously trained feedforward network to the new battery.

The structure of a feedforward network allows us to tune a trained network to the new parameters with partial data because it only uses action parameters as input. Figure 6 shows the forecast for M9 with battery 2 (green), produced by a feedforward network initially trained on data-set M6 (battery 1). The blue line shows the same network tuned with partial data (10% drop in SOC) from M9 (battery 2). The network tuned on partial data produces an improved forecast. Absolute errors for both the forecasts show that the tuned network performs significantly better than the original network. For a recurrent neural network, measured SOC is an input to the network, but data available for tuning only covers 10% of the input space for SOC. We show in Sec. 5.4 that the networks must be trained on the entire parameter space, thus recurrent networks can not be used for partial tuning.

5.4 Parameter-space for Training

Figure 7 and 8 show uniformly downsampled parameters x (denoting length of the edges between waypoints to be traversed in the mission) and θ (denoting angle of turns the robot has to make during the mission), respectively, for each of these missions. Networks trained over data from two missions (M2, M7, marked in red in Figs. 7 and 8) had poor performance compared to networks trained on other missions (c.f. Fig. 9).

Figures 7 and 8 show that the parameter space (distance x and angle θ) for missions M2 and M7 is much smaller in comparison to that of the remaining missions, limiting the learning space for the networks. A network trained on M2 and M7 will only model the input space covered by those data-sets. Thus, for more accurate models, parameters in training missions should be well distributed and adequately cover typical parameter values.

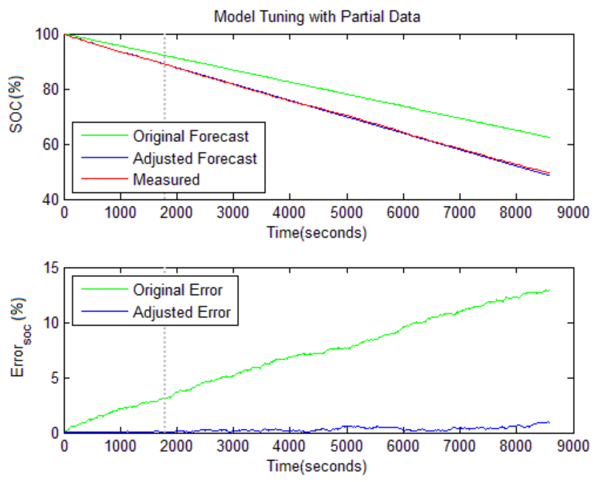


Figure 6: Forecast performance of the feedforward network with and without tuning. Data used to tune the network is marked by a vertical dotted line.

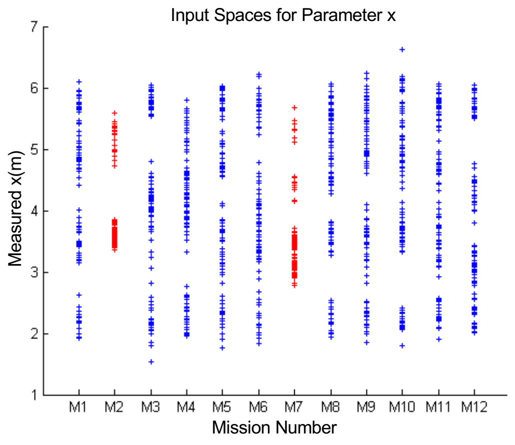


Figure 7: Parameter x for all missions M1-M12.

5.5 Comparison with existing method

We compare the forecasts produced by our framework with that of Parasuraman *et al.* [13], due to the direct relevance of methods used as well as closely related objectives. Figure 10 shows that our method performs significantly better in terms of forecast accuracy; this may be due to their assumption that the power model for locomotion is linear with respect to the average speed (this may be true for smaller robots, but is inflexible). Their additive model also does not account for voltage being shared across all resources, while our model learns from concurrent actions. Acceleration effects were also ignored due to the small size of their robot (the Khepera III is 690 g while the Turtlebot 2 without netbook is 6.3 kg). Thus, our framework is more robust in forecasting SOC for longer missions.

5.6 Alternate Mission Descriptions

The experiments described so far in this section have been performed on missions represented with Mid-level Job-set. Mean absolute errors were computed over entire missions for forecasts produced by networks trained for the three mission representations.

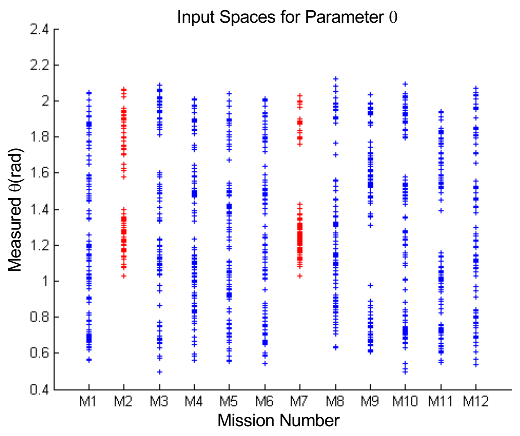


Figure 8: Parameter θ for all missions M1-M12.

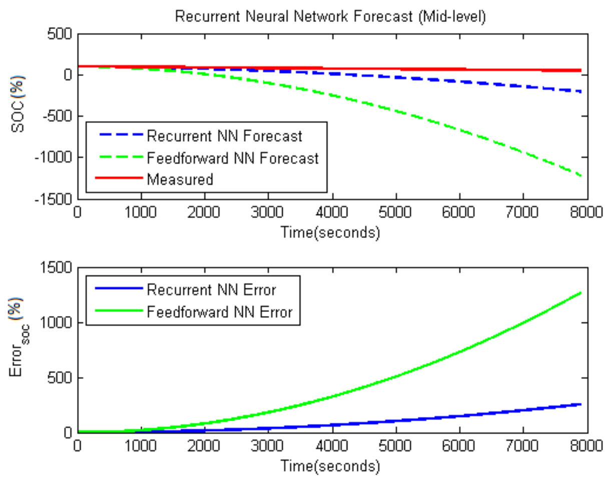


Figure 9: Absolute and relative temporal errors for the two networks.

Figure 5 shows the mean absolute errors for forecasts (10 training-testing data pairs) of the two types of networks for all three mission representations. As the mission representation becomes more compact, the forecast error increases because higher level mission representations encode less information that is relevant to power consumption. Thus, a network trained on more abstract job-sets models the system less precisely, resulting in higher approximation error. Conversely, low-level mission representations encode more precise information regarding the dynamics of power consumption in the execution of the mission. It results in longer mission representations and more data on which a network needs to be trained, but results in lower approximation errors and an overall higher forecast accuracy.

6. CONCLUSION

In this paper, we presented a generalized framework for mission-specific energy awareness in battery-operated robots. We evaluated the framework with three mission representations encoding varying levels of detail. The approach uses recurrent and feedforward neural networks to predict the change in state of charge (SOC) given a mission description. We show that recurrent networks are consistently more accurate predictors of SOC than feedforward networks. However, feedforward networks can be adjusted to new parameters (e.g., a battery with different state of health) by tuning a network learned on different parameters with a small amount of new mission data. Finally, we compared forecasting performance of our work to an existing method in the literature, and showed that our method produces more accurate forecasts.

The work presented is applicable to any type of battery-powered robot. Note, however, that prediction is most effective in predictable environments, and that as discussed with respect to Missions 2 and 7, sufficient training under a similar action space is required.

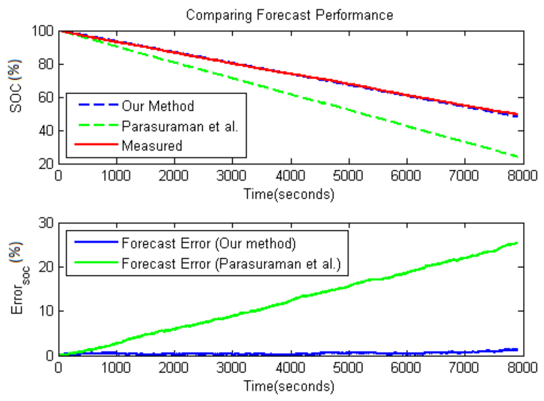


Figure 10: Comparison of forecast performance of our method with Parasuraman *et al.* [13].

7. ACKNOWLEDGMENTS

This work was supported by Office of Naval Research grant N00014-14-1-0734.

8. REFERENCES

[1] S. Abu-Sharkh and D. Doerffel. Rapid test and non-linear model characterisation of solid-state lithium-ion batteries. *Journal of Power Sources*, 130(1):266–274, May 2004.

[2] M. Charkhgard and M. Farrokhi. State-of-charge estimation for lithium-ion batteries using neural networks and EKF. *IEEE Trans Ind Electron*, 57(12):4178–4187, 2010.

[3] M. Chen and G. A. Rincon-Mora. Accurate electrical battery model capable of predicting runtime and iv performance. *IEEE Trans. Energy Convers.*, 21(2):504–511, 2006.

[4] V. De Carolis, D. M. Lane, and K. E. Brown. Low-cost energy measurement and estimation for autonomous underwater vehicles. In *OCEANS*, Taipei, April 2014.

[5] D. Doerffel and S. A. Sharkh. A critical review of using the peukert equation for determining the remaining capacity of lead-acid and lithium-ion batteries. *Journal of Power Sources*, 155(2):395–400, April 2006.

[6] D. D. Domenico, G. Fiengo, and A. Stefanopoulou. Lithium-ion battery state of charge estimation with a Kalman filter based on a electrochemical model. *IEEE International Conference on Control Applications*, pages 702–707, Sept 2008.

[7] F. M. González-Longatt. Circuit based battery models: A review. In *2nd Congreso IberoAmericano De Estudiantes de*

Ingenieria Electrica, Puerto la Cruz, Venezuela, 2006.

[8] V. H. Johnson. Battery performance models in ADVISOR. *Journal of Power Sources*, 110(2):321–329, August 2002.

[9] R. C. Kroeze and P. T. Krein. Electrical battery model for use in dynamic electric vehicle simulations. In *IEEE Power Electronics Specialists Conference*, pages 1336–1342, Rhodes, June 2008.

[10] J. R. LeSage and R. G. Longoria. Characterization of load uncertainty in unstructured terrains and applications to battery remaining run-time prediction. *Journal of Field Robotics*, 30(3), 2013.

[11] N. Mathew, S. L. Smith, and S. L. Waslander. A graph-based approach to multi-robot rendezvous for recharging in persistent tasks. In *IEEE International Conference on Robotics and Automation*, pages 3497–3502, Karlsruhe, Germany, May 2013.

[12] E. Meissner and G. Richter. Battery monitoring and electrical energy management: Precondition for future vehicle electric power systems. *Journal of Power Sources*, 116(1-2):79–98, July 2003.

[13] R. Parasuraman, K. Kershaw, P. Pagala, and M. Ferre. Model based on-line energy prediction system for semi-autonomous mobile robots. In *IEEE International Conference on Intelligent Systems, Modelling and Simulation*, pages 411–416, Langkawi, June 2014.

[14] J. Pentzer, S. Brennan, and K. Reichard. On-line estimation of vehicle motion and power model parameters for skid-steer robot energy use prediction. In *American Control Conference*, pages 2786–2791, Portland, OR, June 2014.

[15] C.-H. Piao, W.-L. Fu, J. Wang, Z.-Y. Huang, and C. Cho. Estimation of the state of charge of Ni-Mh battery pack based on neural network. In *IEEE International Telecommunications Energy Conference*, pages 1–4, Incheon, October 2009.

[16] G. L. Plett. Extended Kalman filtering for battery management systems of LiPB-based HEV battery packs: Part 1. modeling and identification. *Journal of Power Sources*, 134(2), 2004.

[17] A. Sadrpour, J. Jin, and A. G. Ulsoy. Mission energy prediction for unmanned ground vehicles. In *IEEE International Conference on Robotics and Automation*, pages 2229–2234, Saint Paul, MN, May 2012.

[18] Z. M. Salameh, M. A. Casacca, and W. A. Lynch. A mathematical model for lead-acid batteries. *IEEE Transactions on Energy Conversion*, 7(1):93–98, August 1992.

[19] W. Shen. State of available capacity estimation for lead-acid batteries in electric vehicles using neural network. *Energy Conversion and Management*, 48(2):433–442, February 2007.

[20] S. L. Smith, M. Schwager, and D. Rus. Persistent monitoring of changing environments using a robot with limited range sensing. In *IEEE International Conference on Robotics and Automation*, pages 5448–5455, Shanghai, May 2011.

[21] A. Widodo, M.-C. Shim, W. Caesarendra, and B.-S. Yang. Intelligent prognostics for battery health monitoring based on sample entropy. *Expert Systems with Applications*, 38(9):11763–11769, September 2011.

[22] T. Yamazaki, K. Sakurai, and K. Muramoto. Estimation of the residual capacity of sealed lead-acid batteries by neural network. In *IEEE International Telecommunications Energy Conference*, pages 210–214, San Francisco, CA, October 1998.