An Artificial Circadian System for a Slow and Persistent Robot*

Matthew J. O'Brien and Ronald C. Arkin

Mobile Robot Lab, Georgia Institute of Technology, Atlanta, GA 30308

{mjobrien,arkin}@gatech.edu

Abstract. As robots become persistent agents in natural, dynamic environments, the ability to understand and predict how that environment changes becomes more valuable. Circadian rhythms inspired this work, demonstrating that many organisms benefit from maintaining simple models of their environments and how they change. In this work, we outline an architecture for an artificial circadian system (ACS) for a robotic agent. This entails two questions: how to model the environment, and how to adapt robot behavior based on those models. Modeling is handled by treating relevant environment states as time series, to build a model and forecast future values of that state. The forecasts are considered special percepts, a prediction of the future state rather than a measurement of the current state. An ethologically-based action-selection model incorporates this knowledge into the agent's decision making. The approach was tested on a simulated precision agricultural task - pest monitoring with a solar powered robot - where it improved performance and energy management.

1 Introduction

Moving from structured, static environments to natural, dynamic environments remains a challenge in robotics. While reactive approaches in behavior-based designs have shown effectiveness in dealing with the randomness present in the real world, most ignore the somewhat predictable dynamics that many environments exhibit. In this work we describe an architecture for modeling environmental dynamics, and adapting an agent's behavior to them. This approach is inspired by circadian rhythms found in a variety of natural organisms; from humans, to plants, to bacteria [15].

Circadian systems are chemical oscillators, synchronized to the solar cycle, that influence behavior and metabolic processes of an organism. Rather than simply reacting to the environmental changes, circadian systems allow an organism to take action before its senses (perceptual state) or needs (internal state) could drive the behavior. "Circadian rhythmicity of behavior represents an animal's information, or one is tempted to say, knowledge about a particular feature of its environment… and what to do about it." [14]

* This research is supported by The Office of Naval Research Grant #N00014-15-1-2115.

The need for such an architecture in a robotic system is growing. Recently, the non-industrial robotic market overtook the industrial robotic market for the first time [18]. With the influx of robots into our lives, interest and research into persistent autonomy is increasing. Particular focus has been placed on mapping dynamic environments. While some model those dynamics [1], many focus on ways to filter changes to update the map [7]. More relevant is work on modeling environment dynamics that is applied to robot behavior. The most similar work modeled binary environment states as a periodic probability using Fourier series, and applied it to several problems including path planning, localization, and exploration [11].

This work is particularly targeted for slow, persistent robotic agents. Persistent agents will experience environmental cycles that a robot that executes for twenty minutes, or even two hours, will not. Slow robots are inherently less able to react quickly to a changing environment (but may have several advantages, especially for persistent energy-constrained tasks [3]). Thus, the ability to predict and proactively act could be very beneficial.

In this research, we develop an artificial circadian system (ACS) to learn and exploit the patterns that often exist in the dynamics of natural environments. In previous work, we demonstrated the application of time series modeling of an environment for a robotic agent, and tested it on a simulated robot interacting with pedestrian traffic using very simple behavioral rules [13]. Here, we investigate a principled way to incorporate the generated predictions into action-selection.

2 The Architecture

The ACS is built on a behavior-based architecture. A special set of perceptual schemas (or perceptual algorithms that process sensor data for specific behaviors [2]) model the environmental dynamics as time series and forecast future values of the state. Behaviors are supplemented with activation functions based on both the current sensed state and the future predicted state. Action-selection is done by selecting the behavior with the highest activation. Figure 1 shows the top-level architecture, and the rest of this section details the components.

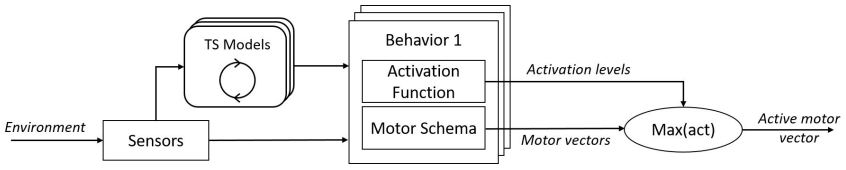

Fig. 1. The top-level diagram of the ACS architecture.

2.1 Modeling environmental dynamics as time series

We view the environment state as a time series, and apply methods from the time series literature to model and forecast future values. Time series models predict, or "forecast", future values of a variable directly from its past values. It allows for mod-

eling of both cyclic and non-cyclic effects, and can forecast both the value of the future state as well as generate prediction intervals. A useful quality is that the approach is data driven. The underlying cause of the dynamics may be too complicated to model, or at least impractical to do so on a robot (consider the actions of many individuals to create traffic). Time series modeling offers versatility and simplicity, side-stepping the need for expert knowledge about a domain.

A fundamental approach to modeling time series is the classical decomposition [8], of a time series into its major components:

$$Y_t = T_t + S_t + E_t \tag{1}$$

Where Y_t is the original time series. T_t is the trend component, a slowly changing average level; S_t is the seasonal component, a repeating pattern with known period; and E_t is the residual, or error, left over after the trend and seasonal components have been removed from the time series. This decomposition provides a structured approach for modeling and allows for focus on components of interest. For example, seasonal effects are sometimes removed from data in a process called deseasonalization. In our work the seasonal effects are of key importance, as they represent the cyclic (or circadian, if the period is roughly 24 hours) dynamics in the environment.

The literature on time series analysis and forecasting provides a broad set of tools [9]. At this time, however, there is no catch-all method or system. Historical data is required to manually select an appropriate model, one that can capture the dynamics of the environment. While model selection is must be done offline, model fitting (i.e. parameter estimation) and forecasting can be done online, autonomously by the agent.

Time series modeling has traditionally been applied for offline analysis of discrete data. To leverage these techniques on a real-time system, three data structures are used when updating the forecast. The past state values, or the measured time series, is stored as the vector S_p. These are fed into the time series model, and used to forecast the future values of the state, S_F^t, where t is the time this forecast vector was generated. Lastly, there is a set of current sensor measurements called S_M. At the end of any forecast interval, S_M is used to generate a final state value (S_p) for that interval. This could be an average of multiple measurements, a count of recorded events, or even a null value if no measurements of the relevant state variable were made.

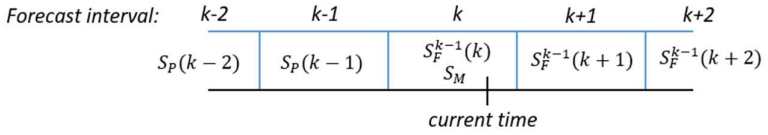

Fig. 2. S_P returns measures of the state in the *past*, while S_F returns forecasts of the state in the future. S_M is the current measured state.

Entrainment of this system to the environment happens at two levels. Every forecast interval the agent appends a new value to S_P, and generates a new forecast vector S_F. This means the forecast for some specific point in time in the future is repeatedly updated (usually becoming more accurate) as new measurements of the environment

are taken. Secondarily the new recent history, S_P, can be used to re-estimate the model parameters. Adapting the time series model if the dynamics of the environment have changed.

2.2 Action-selection via activation levels

The action-selection method used in this work is part of a family of approaches that assigns an activation level to each top-level behavior an agent can execute. Arbitration via action-selection is straightforward: the behavior with the highest activation level is executed. This leaves the question of how to calculate that activation level. Many approaches for this problem have been developed [16, 5, etc.]. The approach in this paper is based on an ethologically-inspired architecture, described in [4].

Every behavior includes several additional components to facilitate the action-selection process. The first is a motivation function (MOT) which represents a behavior's internal motivation to activate. This is based on endogenous (internal) variables representing the state of the agent (such as its power level). The second component is the releasing mechanism (RM). It differs from the motivation function in that it focuses on external, or exogeneous, variables. The RM is a special function that specifies both the conditions required for a behavior to activate, and how well they are satisfied. Algorithmically, if its output is zero it means necessary conditions to execute a behavior are not currently satisfied, and the behavior cannot execute regardless of how high its total activation may be. In this research, we introduce a new component, the circadian rhythm function (CIR). This component represents the influence of an associated forecast on a behaviors activation level. Like the releasing mechanism, the circadian rhythm function focuses on exogenous variables, but considers the impact of their future predicted values, rather than their current measured value.

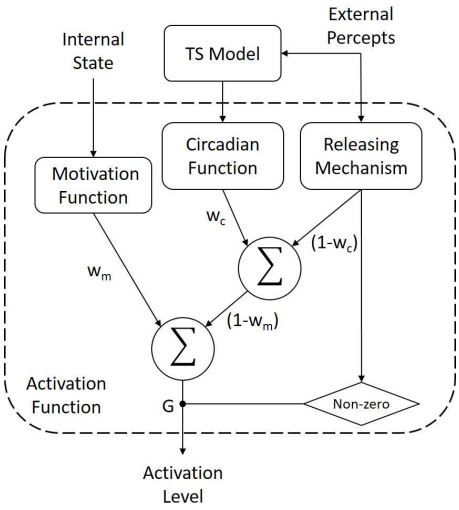

Fig. 3. The activation function of a behavior is a weighted sum of the three components, plus a conditional check on the releasing mechanism

Each behavior has an activation function that is the weighted sum of the motivation function, releasing mechanism, and circadian rhythm function. The activation function produces a final activation level used to select which behavior to execute. The activation function of behavior N is:

$$act_N = \begin{cases} 0 & if\ RM\ conditions\ not\ satisfied \\ G * \left(w_m * MOT_N + (1 - w_m)(w_C * CIR_N + (1 - w_C) * RM_N)\right) & else \end{cases} \quad (2)$$

Three parameters are introduced to conveniently adjust robot behavior, based on the approach taken in [6]. The behavior gain, G, adjusts the overall activation magnitude of a behavior. The weight $\{w_M \in R \vee 0 \leq w_M \leq 1\}$ adjusts the relative influence of the releasing mechanism and the motivation function. This allows for the robot to become more reactive to its environment (higher weighting of releasing mechanism and circadian rhythm function) or more focused on its internal needs and goals (higher weights of motivation function). A final weight is introduced, $\{w_C \in R \vee 0 \leq w_C \leq 1\}$. This weight balances the impact of the CIR function against the releasing mechanism, or more generally sets how much the robot focuses on the current measured state compared to the predicted future state.

The circadian function allows for more than acting directly on a forecasted state. For instance, choosing an ideal time to execute a behavior over some time window can be done by comparing the current state against the forecasted state throughout that window. An otherwise hidden state can be estimated using forecasts generated in the past of the now current state. In the following experiment, both of these approaches are leveraged.

3 Experimental Validation

To validate this architecture, we set up a simulated experiment to test how incorporating the circadian rhythm function impacts an agent's behavior. The scenario involves a small, slow agriculture robot; one that persists within the agricultural environment as a beneficial component of the ecosystem, monitoring and tending to the plants.

The agent's purpose is to monitor the population of a pest over a growing season, and identify when the population reaches a critical threshold requiring intervention by a farmer. It must do so while spending a minimal amount of time and energy, allowing the agent to work on other tasks, and reducing wear on physical components. The dynamics modeled by the time series will be the pest population and solar energy. This will inform when the agent must spend more time and energy on monitoring the pest population, and when it should charge.

3.1 Environment Details and Modeling

The simulation is implemented in Gazebo as shown in Figure 4. The pest population dynamics follow a model based on aphids. Many aphid species exhibit exponential population growth after an initial infestation due to their ability to perform rapid asexual reproduction during some periods of their life cycle. This growth is

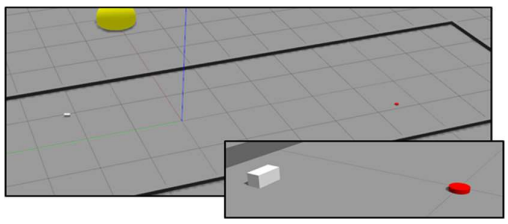

Fig. 4. The Gazebo simulation environment. The bottom right image highlights the robot (white box) and pest (red disk)

simulated by inserting pests according to a simple aphid population model [10], plus a random noise factor. Pest are distributed uniformly over the work space. Figure 5 shows several simulated trajectories of the aphid population. The robot is solar charged, and must spend a significant portion of its time charging. It expends energy whenever moving, proportional to its speed. The robot receives "direct" solar light when against the top wall closest to the "sun" (yellow sphere), and reduced solar light as it moves away. Available solar power over time followed a bell shape plus noise between dawn and dusk, and was zero at night time.

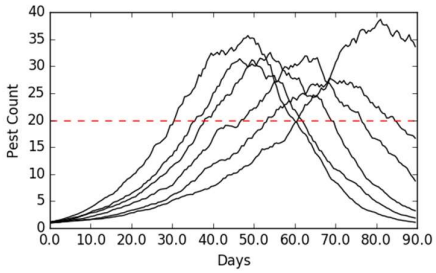

Fig. 5. Example population trajectories used for simulation. The critical threshold for pest population, where intervention is needed, is shown as a dashed red line. Different trials reached this threshold over 30 days apart.

The pest dynamics were stored as a time series of measured pest levels, defined as the number of pests found per area searched. A 12-hour time interval was used. As the agent often did not monitor in a 12-hour period, values between measured intervals were interpolated, while values after any measurement were forecasted. The pest dynamics were modeled using the STL decomposition (Seasonal and Trend decomposition using Loess) in the R forecasting library [9] which breaks the series into a trend, seasonal, and stationary components. For details on these forecasting techniques, see [8]. Nine previous seasons, each 90 days long, were simulated to provide data to fit the model parameters.

Solar energy is this simulation was simple, a set cyclic function over 24 hours, plus noise. Thus, it was forecasted as the average or expected solar level per time step.

3.2 Robot Behaviors

While the top-level action-selection is handled by the architecture described in section 2, each behavior is built as an assemblage of simpler behaviors. The behaviors are based on work exploring the design of slowbots (i.e. slow robots [17]). The monitor behavior is responsible for pest monitoring. It drives the robot around its work space, exploring and searching for pests. This is implemented as a vector summation of three behaviors: a wander behavior to drive the robot around its work space, an attachment behavior [12] to keep the robot in its workspace, and an obstacle avoidance behavior.

For the activation function, the monitor behavior's MOT (equation 3 below) drives the agent to monitor the environment at periodic intervals. When not monitoring, its

activation level increases proportional to the time since it last monitored. When monitoring, the activation level decreases proportional to the time is has been recently monitoring (we used a 12-hour window). The RM (equation 4) increases activation of the monitor behavior based on how much the forecasted pest level, and the last measured pest level, disagree. This causes the robot to monitor longer and more frequently when the current pest population deviates from expected values. The RM has a small constant factor to always release the monitor behavior, as the environment is always available to monitor. The CIR (equation 5) increases activation proportional to the expected level of pests. This is the key component that prioritizes the robot's actions when there are more pests, or a higher expected number of pests.

In the below equations, T_S is the time (minutes) since the monitoring behavior last executed. T_M is the time (minutes) spent monitoring in the last 12 hours. P_M is the last measured pest level, and $P_F(t)$ is the forecasted pest level for time t. All K_n values are constant parameters.

$$MOT(\,T_M, T_S\,) = \begin{cases} K_1 * T_S & if\ monitor\ inactive \\ 100 - K_2 * T_M & if\ monitor\ active \end{cases} \tag{3}$$

$$constrained\ such\ that \quad 0 \le MOT(\,T_M, T_S\,) \le 100$$

$$RM(\,P_M, P_F(t)\,) = K_3 * |P_M - P_F(t)| + 0.1 \tag{4}$$

$$CIR(\,P_F(t)\,) = K_4 * P_F(t) \tag{5}$$

The energy collection behavior moves the robot to the region of its work space where direct sunlight is available. Once there the agent waits, charging, until another behavior takes over executing. This is implemented through a simple FSA (see Figure 6).

The energy collection behavior's motivation function (equation 6) is a function of the internal battery level, and it increases exponentially as the robot's battery level decreases. This was arbitrarily chosen, but ensures the behavior becomes dominant when power is low. The RM (equation 7) increases the activation level proportional to the amount of solar energy currently available to the robot, averaged over the recent history. The behavior is released even when there is no sunlight so that the agent can move and wait for energy, rather than monitor when its battery might be too low. Finally, the CIR (equation 8) looks at the average solar energy for a set time in the future (in this experiment, 90 minutes), and compares that to the current solar energy. When the current solar energy is higher, the activation level of energy collection is increased, making the robot more likely to charge before the opportunity is lost.

In the following equations, B is the battery level of robot (as a percentage), E_M is the current measured solar energy, and $E_F(t)$ is the forecasted solar energy at some point in time, t.

$$MOT(\,\mathrm{B}\,) = K_5 * 10^{1-B} - K_6 \tag{6}$$

$$RM(\,E_M\,) = K_7 * E_M + 0.1 \tag{7}$$

$$CIR\big(E_M, E_F(t)\big) = \begin{cases} K_8 * \big(E_F(t) - E_{avg}\big) & if\ E_F(t) - E_{avg} > 0 \\ 0 & else \end{cases} \tag{8}$$

$$E_{avg} = \frac{1}{N} \sum_{i=t}^{i=t+N} E_F(i) \tag{9}$$

The final behavior, rest, simply causes the robot to stop moving. This behavior exists as an alternative to always attempting to collect energy or monitor, even when the activation of both behaviors is low. Thus, if there is nothing important for the agent to do, the agent will do nothing. The agent will still charge if its positioned in direct sunlight while resting. The rest behavior maintains a constant activation level at all times, achieved by a constant value for the MOT, RM, and CIR functions. There is potential for a more sophisticated activation function, potentially taking into consideration over heating or bad weather, but this was not the focus of this work. Figure 6 shows the top-level diagram of the robot's architecture for this experiment.

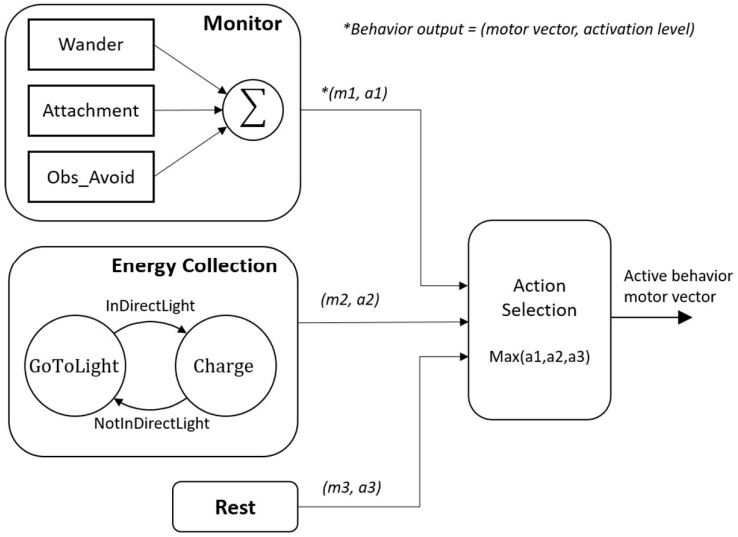

Fig. 6. Top-level behavioral architecture

3.3 Experiment Procedure

The robot's task was to detect when the pest population crossed a critical threshold. This goal was achieved by executing the monitoring behavior and measuring the pest population after said threshold is reached. The experiment varied the weight of the CIR function for both behaviors (monitor and charge) across a range of values. For performance with respect to the goal, the *time-to-detection* measure is defined as the time between when the critical threshold was reached, and when the robot detected it. With respect to energy, two others are defined. *Energy-spent* is the energy used by the robot in one trial. *Charge-rate* is the rate of energy collected, per minute, during the

periods the robot was charging. These measure the efficiency of both working and charging in terms of energy.

The experiment was executed 20 times for each condition. Trials had different initial conditions, pest locations, and pest population growth included some randomness. The hypotheses for the experiment was that the inclusion of the CIR component in the activation function of the monitor behavior will cause the robot to detect that the pest population reached the threshold sooner (lower *time-to-detection*) while spending less time/energy in the monitoring behavior (lower *energy-spent*). For the charge behavior, the CIR component will improve the timing of when the robot charges (higher *charge-rate*).

4 Results and Discussion

4.1 The Monitor Behavior

When testing the monitor behavior, the charge behavior weights were set (by experimentation) to: $W_M = 0.5$, $W_C = 0.3$. The monitor activation weights were $W_C = 0.9$, and W_M varied from 1.0 (only use MOT) to 0.4 (heavily use CIR and RM components). Figures 7 and 8 show the performance for *time-to-detection* and *energy-spent*. There is a clear trend that higher weighting of the CIR component reduced the average *time-to-detection*, reducing it by **85.9%** between 0% and 54% CIR weights (2267 minutes to 319.9). The time-to-detection also has significant variance, as the robot could randomly monitor right before or after the pest threshold is reached. The standard deviation was reduced by **74%** with higher weighting of the CIR: from 1352 minutes to 351.

Figure 9 illustrates the difference in behavior. Without the CIR component, the agent uses no knowledge about the pest population, and monitors at regular intervals. In this case, the ACS predicts an otherwise hidden variable: the level of pests when the robot is not monitoring. This allows the agent to use less energy to monitor when the pest threat is low, but more energy to monitor near the critical time when pests are

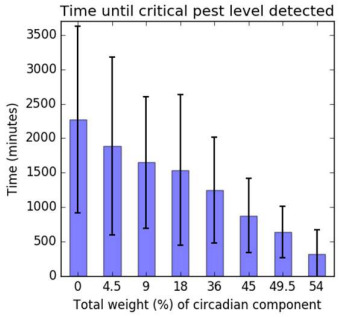

Fig. 7. Average *time-to-detection* over 20 trials when monitor behavior weights varied. One standard deviation marked.

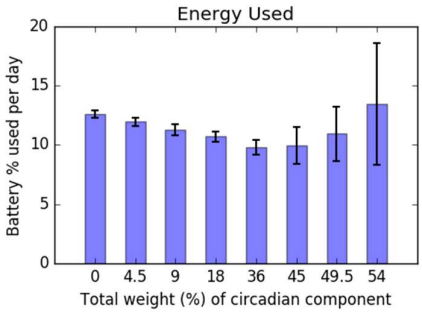

Fig. 8. Average *energy-spent* per day over 20 trials when monitor behavior weights varied. One standard deviation marked.

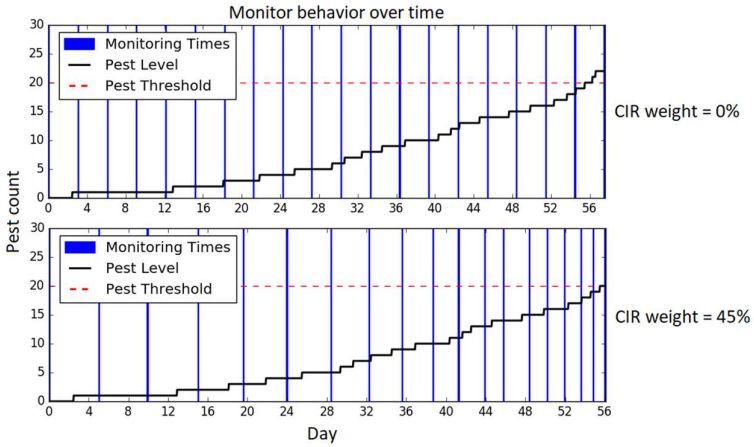

Fig. 9. Two executions by the agent on the same trial with different activation function weighting. This highlights the difference between when the agent monitors over the course of the experiment if the CIR component is included in the activation function.

becoming a problem. The average energy used per day was minimized at 36% CIR weight, where it was **22.2%** lower than with 0% CIR weight. With higher CIR weight, the overall energy usage begins to increase, up to **6.71%** above the value at 0% CIR weight. Once the CIR weight reaches high enough, the agent may begin to monitor continuously (if it has energy to do so). This depends on the forecasted pest level, it did not happen in every trial, which creates the rapidly growing variance in energy usage. This effect happened when the CIR component alone can surpass the rest behavior in terms of activation. This significantly reduces the average *time-to-detection*, but expends a large amount of energy to do so.

4.2 The Charge Behavior

When testing the charge behavior, the monitor behavior was held constant with weights $W_M = 0.5$ and $W_C = 0.9$. The charge behaviors weights were adjusted with a slightly different scheme. The releasing mechanism was an important component to avoid behavioral dithering (or rapid switching). It was left at a total 30% of the activation weight, while the total weight of the CIR component was varied from 0% to 40%. W_M and W_C were both varied to achieve these ratios of the MOT and CIR components on final activation level.

The *charge-rate*, as shown in Figure 10, has a trend of increasing with greater weight of the CIR component: from an average 0.1778 charge-% per min to 0.4187, an increase of **135%**. The charge behavior's circadian rhythm function particularly prioritizes charging near peak sunlight, before the ideal opportunity to charge is lost. Thus, the agent is reacting not just on its current power and the current solar energy, but also on how the solar energy is changing. This has a clear impact on performance. The *time-to-detection* was also investigated for the charge behavior testing

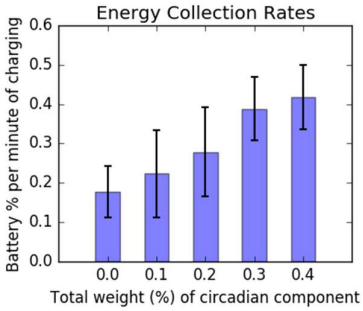

Fig. 10. Average *charge-rate* over 20 trials when charge behavior weighs varied. One standard deviation marked.

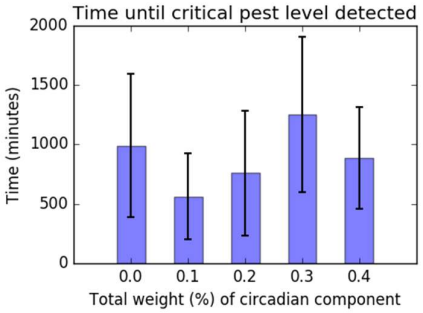

Fig. 11. Average *time-to-detection* over 20 trials when charge behavior weights varied. One standard deviation marked.

(Figure 11) to see if there might be effects on performance due to the changes to charge timings. However, no trend seems to have emerged.

5 Conclusions

This work has detailed an architecture to help a robotic agent respond to long-term dynamics - the Artificial Circadian System. Inspired by the circadian systems in nature, time series models forecast the changing environment. These forecasts are incorporated into the action-selection process to allow a robot to act based on both the current measured and future predicted values of the state. The time series models can also be applied to estimate an otherwise hidden state, or detect deviations in the environment by comparing an expected and measured state.

An agricultural domain was chosen as a test bed, and a pest-monitoring task with a solar-powered robot was simulated. With the ACS, the agent was able to model the pest level over time, even when not actively monitoring it. Results showed this allowed for quicker identification of when the pest level reached a critical threshold requiring action, while using less energy to do so. The agent also weighted the future predicted availability of solar energy into its decisions on when to charge, improving the timing of transitions to a charging behavior. The agent's behavior adapted to the changing environment: monitoring more when the pest level was higher, and prioritizing charging when solar energy was at its peak.

Future work is planned in several directions. The next step is to leave simulations and begin testing with physical robots interacting with real environments. Robustness should be developed to handle cases when the forecasting system fails, due to modeling error or random events, allowing the agent to detect and respond to these situations. Finally, reinforcement learning offers interesting possibilities for the agent to autonomously adapt even when the environmental dynamics themselves change, but also brings significant challenges. A single state becomes a multi-dimensional trajectory when forecasted, significantly increasing the complexity of the state space.

References

1. Ambrus, R., Ekekrantz, J., Folkesson, J., Jensfelt, P.: Unsupervised learning of spatial-temporal models of objects in a long-term autonomy scenario. In: Intelligent Robots and Systems (IROS 2015), pp. 5678-5685. Curran Associates, Inc (2016)
2. Arkin, R.C.: Behavior-based robotics. MIT press. (1998)
3. Arkin, R.C., Egerstedt, M.: Temporal heterogeneity and the value of slowness in robotic systems. In: IEEE International Conference on Robotics and Biomimetics (ROBIO 2015), pp. 1000-1005. Curran Associates, Inc (2016)
4. Arkin, R.C., Fujita, M., Takagi, T., Hasegawa, R.: An ethological and emotional basis for human–robot interaction. Robotics and Autonomous Systems. 42(3), 191-201 (2003). doi: 10.1016/S0921-8890(02)00375-5
5. Blumberg, B.: Action-selection in hamsterdam: Lessons from ethology. In: Proceedings of the Third International Conference on Simulation of Adaptive Behavior: From Animals to Animats, pp. 108-117. MIT Press (1994)
6. Chernova, S., Arkin, R.C.: From Deliberative to Routine Behaviors: A Cognitively Inspired Action-Selection Mechanism for Routine Behavior Capture. Adaptive Behavior, 15(2), 199-216 (2007). doi:10.1177/1059712306076255
7. Dayoub, F., Duckett, T.: An adaptive appearance-based map for long-term topological localization of mobile robots. In Intelligent Robots and Systems (IROS 2008), pp. 3364-3369. Curran Associates, Inc (2009)
8. Hyndman, R., Koehler, A.B., Ord, J.K., Snyder, R.D.: Forecasting with exponential smoothing: the state space approach. Springer Science & Business Media (2008)
9. Hyndman, R.J.: Forecasting Functions for Time Series and Linear Models. R package. http://pkg.robjhyndman.com/forecast/ (2017) Accessed Jan 2018
10. Kindlmann, P., Arditi, R., Dixon, A.F.G.: A simple aphid population model. In: Aphids in a New Millennium, pp. 325-330. INRA, Paris (2004)
11. Krajník, T., Fentanes, J.P., Santos, J.M., Duckett, T.: Fremen: Frequency map enhancement for long-term mobile robot autonomy in changing environments. IEEE Transactions on Robotics. 33(4), 964-977 (2017). doi:10.1109/TRO.2017.2665664
12. Likhachev, M., Arkin, R.C.: Robotic comfort zones. In: Sensor Fusion and Decentralized Control in Robotic Systems III, International Society for Optics and Photonics, vol 4196, pp. 27-42 (2000). doi:10.1117/12.403722
13. O'Brien, M.J., Arkin, R.C.: Modeling Temporally Dynamic Environments for Persistent Autonomous Agents. In: Proceedings of the Thirtieth International Florida Artificial Intelligence Research Society Conference, pp. 442-448. The AAAI Press, Palo Alto (2017)
14. Oatley, K.: Circadian Rhythms and Representations of the Environment in Motivational Systems. In: McFarland, D.J. (eds.) Motivational Control Systems Analysis, pp. 427-459. Oxford: Academic Press. (1974)
15. Paranjpe, D.A., Sharma, V.: Evolution of temporal order in living organisms. Journal of Circadian Rhythms. 3(7), (2005). doi:10.1186/1740-3391-3-7
16. Richter, M., Sandamirskaya, Y., Schöner, G.: A robotic architecture for action selection and behavioral organization inspired by human cognition. In: Intelligent Robots and Systems (IROS 2012), pp. 2457-2464. Curran Associates, Inc (2012)
17. Velayudhan, L., Arkin, R.C.: Sloth and Slow Loris Inspired Behavioral Controller for a Robotic Agent. IEEE-ROBIO, Int. Conf. on Robotics and Biomimetics, Macau, (2017)
18. The Robotics Industry Will Reach $237 Billion in Revenue Worldwide by 2022. Tratica. https://www.tractica.com/newsroom/press-releases/the-robotics-industry-will-reach-237-billion-in-revenue-worldwide-by-2022/ (July 5th, 2017) Accessed January 2018