DISSERTATION

ANTICIPATION ENHANCED BEHAVIOR-BASED ROBOTICS USING INTEGRATED SYSTEM DYNAMICS

Submitted by

Douglas A. Hopper

Department of Mechanical Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Spring 2017

Doctoral Committee:

Advisor: Wade O. Troxell

David G. Alciatore
Paul R. Heyliger
Lou B. Bjostad

ABSTRACT

ANTICIPATION ENHANCED

BEHAVIOR-BASED ROBOTICS USING

INTEGRATED SYSTEM DYNAMICS

Behavior-based robotics specifies behavior as the interaction between the task, environment, and agent with specific capabilities that creates a successful behavior to attain task achievement. Observed task achieving behavior is confirmed and validated by a prespecified performance criteria. For behavior-based robotics, conditions in the niche environment are directly matched to and cue behavior choice that yields task achievement by the robot agent. A minimalist approach attains this behavior choice from only a few possible scenarios for the niche environment and a simple associated response. Previous work in behavior-based robotics has been generally limited to a reactive response to environmental conditions, with little or no notion of looking ahead to potential successful future outcomes.

Focus on the notion of anticipation provides a novel addition to the task achieving behavior-based robotics approach. Anticipation is the formulation of suitable processes to manifest behavior from a small set of feasible scenarios in the near future before the outcome is certain. Anticipation results in successful behavior beyond mere reactions to niche conditions that leads to desired task achievement with expected perceived immediate or later reward based on suitable fitness matched to the niche.

The approach to add anticipation developed a formal system dynamics model to represent previously known behavior archetypes, extended them with the notion of anticipation, and enhanced the system dynamics operation. Simulation of a robot instance using anticipation for wall following, called the TOURIST, was conducted to gain insight into behaviors that would be observable in a real world natural system. Simulation of the TOURIST robot with anticipation built into the archetype programming illustrated the advantages of including the notion of anticipation. Anticipation allows a TOURIST robot agent to travel a smoother path and make choice of small increments in behavior change that produce more desired longer term responses. With anticipation, numerous small adjustments are made that require less energy than large spins of the SEEK behavior, so only one third of the SEEK behaviors occur, and thus wastes less energy and time. Also with anticipation, the TOURIST makes twice as many cycles of the area at the same speed and in the same time, so a broader range of area is covered and can more readily perceive any dynamic changes in the overall arena. The methods and insights were added to a real world robot instance, and the benefits of anticipation were observed to occur. A specific metric, ANNum, was developed for describing operation of the TOURIST robot. Greater metric values were found with anticipation on, reflecting more behavior responsiveness to the niche per unit time when anticipation was used.

In conclusion, anticipation enhances robotic performance by manifesting task achieving behavior that is properly matched to a specific niche condition. Anticipation extends beyond the merely reactive behavior previously used in behavior-based robotics by acting like a funnel or channel to guide the behavior choice to match a specific niche. The observed behavior choice is manifest *before* the outcome is realized and certain to occur. As a practical result, the robot agent is able

to make many smaller adjustments earlier and faster with better chance for desired outcome than would be observed without anticipation. It circumvents repeated larger adjustments that waste more resources and take more time for task achievement. Such enhanced anticipation behavior avoids obstructions and potential destructive paths or motion, and is more able to achieve tasks such as to find objects and move along walls with minimal effort. Thus, anticipation that is added to robot architecture improves behavior choices to realize desired task achievement. Future work could add anticipation to real world practical automation and robotics to further test the improved operation with anticipation. In summary, anticipation observed in a robot agent should act before the outcome is known, make timely small adjustments toward a goal, and appear as if the future were known ahead of time.

ACKNOWLEDGEMENTS

This work culminates several challenging and thought provoking years reflecting the support of academic colleagues, advisors, counsel, friends, and family members that I consider part of my team. The dialog developed between various team members has led to the creation of this dissertation. I wish to thank my advisor, Dr. Wade Troxell, for the many hours we have met together to discuss the root concept of this dissertation: anticipation. Also, many thanks to my committee members, Drs. Dave (Dr. Dave) Alciatore, Paul R. Heyliger, and Lou Bjostad for the courses I shared with them and the many concepts they introduced me to. Many other friends and colleagues are too numerous to mention, but a few that are most memorable are Joe Wilmetti (who provided practical help , Dave Knight (who presented the template for the Dissertation), Dr. Dave Alciatore (again, for having me as GTA for the Mechatronics course multiple times), Mike Kostrzewa (for many years of support via Industrial Assessment Center work experience), John Waddell (for long walks and discussions to clarify situations, and CSU volleyball diversions), Jim Ells (for the Saturday breakfast), Ihsan Abbud (for focus on daily AVF activities), Paul and Gail Hein (for holiday hikes), Jeff Lemke (for making the hikes interesting), Coach Tom Hilbert (CSU Volleyball coach who endued many questions as to how competitive encounters work), the CSU Volleyball team (for providing a needed diversion at many times), Dave D'Alessandro (for discussing how to resolve issues), Wayne Viney (the Serendipity Sunday School class leader with broad discussions often involving anticipation), Serendipity Class (many various opinions to consider), and other friends that listened to discussion of robotics or concepts of anticipation to garner inspiration and aid in formation of the concepts and ideas.

Funding support was provided through many work activities, including Alan Kirkpatrick having me rework experiments in the CSU Thermosciences Lab, Mike Kostrzewa directing the Industrial Assessment Center visits to small manufacturing plants along with Energy Efficiency Assessments in Agriculture that matched my background skills, GTA positions in the Mechanical Engineering department both in the Thermosciences and Mechatronics Lab, and my apartment environment where I worked both as Groundskeeper and Community Pride Coordinator that allowed for numerous horticultural and sustainable interactions with myriads of persons across cultures.

Of course, the Love and direction of my family is especially appreciated. Thank you to my Mom, Janet A. Hopper, for her unfailing love and hours of phone conversations. Thanks to my Dad, Clayton L. Hopper, for his love and support, and for his early incorporation of a work ethic with Mom) that gave the perseverance to complete such a task, though he is now passed. Thanks to my brother, James C. Hopper, and his family for having been a continuous companion while growing up, and one who shares my appreciation for the Homestead we grew up in. We have share the good times and the bad, and look forward to many more good times.

And a Universal Thank You to God for direction and grounding in Faith that someday we will all share in abundant reward. Anticipation and its benefits to Life have been imparted to us all through the efforts of the Creator that we revere, respect, and Love.

DEDICATION

To my family: may they anticipate and achieve continuing success in the future.

Yet, in all things, consider this:

> "Future contingencies that have no implications for present commitment have no relevance to design."
>
> Herbert Simon, in: The Sciences of the Artificial (1996).

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

OVERVIEW OF THE PROBLEM

Focus of the Study

Anticipation is the focus of this study, and how it can be included in robotics. The intent is to understand the entire notion of anticipation, from what it is, to how it acts, to how it is recognized, to describing a congruence framework, to defining means and methods that include anticipation in robotic architecture to effect behavior and realize task achievement.

Robotic Systems

Robotics is a rapidly growing field. It includes aspects of automation, traditional classical artificial intelligence (AI), newer behavior-based robotics (BBRs), human in the loop (HIL) and hybrid systems with aspects across defined areas. Approaches vary from that of minimalist that perform only necessary operations, to complex sophisticated planning algorithms that define a myriad of problem solutions. Control systems vary from open loop (no feedback for error control), to feedback control that may include components of proportional, derivative, and integral (PID) algorithms, to neural net learning type algorithms. Systems themselves may be closed systems, somehow isolated from the outside world, to open systems that continuously receive input from and provide output to the surrounding environment. Form of robots is as simple as devices such as a washing machine with a specific purpose, to more complex Mars rovers that must cope with uncertain environments, to humanoid robots that are intended to work with humans to provide daily assistance and have potential for social interactions. Aside from

understanding the broad range of definitions and applications in robotics, a contribution to robotics necessitates one to focus on some area in which to provide novel advancements.

Behavior-Based Robotics Architecture Approach

Behavior-based robotics (BBRs) is defined as an approach that matches robot agent behavior directly to a specific environmental niche condition to promote desired task achievement. A robot agent can be a specific physical robot or programed process that performs a behavior to attain desired task achievement. The minimalist BBRs approach provides ample opportunity for study and advancement of robotics as an engineering discipline. Overall, one BBR principle is that there is no general purpose robot, but instead a robot is designed to perform a specific task in a specific environmental niche context, and that yields desired observed behavior. For early instances, a BBR agent (robot) reacted directly to specifics in the niche to create behavior, thus operating in a reactive mode. Outcomes followed the traditional view of science that causation operates only in one direction, so that all future events are based entirely on direct causation from certain specific past events. A current state is dictated specifically by a set of past states, so the future is always known based on the past. This view aligns with traditional disciplines of physics and chemistry, the hard sciences, where purpose and human emotions are ignored. More soft sciences such as psychology, philosophy, and theology take a less strict world view. Biochemistry as a discipline embraces a less structured causality with pathways that include probabilistic stochiometric equation relations. All of the life sciences embrace the scientific method that seeks to remove human judgement from the determination of truth, instead basing the determination on results from strict experimentation. Here biology shares the reductionist approach with physics and chemistry, contending that all causality can be broken down into

components to be further understood, and then reassembled to find the overall truth for actions of a system.

Systems Approach Studies

On the other hand, the general systems and wholistic approaches vehemently argue that the system is much more, and indeed quite different, than the sum of its parts. Dismantling the system to study it destroys the very thing of interest that generates the observed behavior: interrelations among the connected essential components. The most important and valuable aspects of the system are embedded in the structure that maintains proper relations, so must be studied as a whole. Even studies of model systems, such as 'lower' animals in biology, are not sufficient to understand the interrelationships important to complex human biological systems, or as represented more broadly in society and economics. To understand internal system dynamics, one is persuaded to turn away from reductionism as the source of all truth, and instead study the entire system to understand nuances and dynamics that affect observed behavior. Modeling of whole systems requires techniques that capture the archetypal causes for various types of behavior in an architecture that includes traits such as time delays and looping pathways that cause positive reinforcement even to the brink of instability, or balancing elements that keep the entire system stable for a range of expected conditions.

Anticipation Architecture

Though the previous BBR approach is reactionary in nature and concentrates on response to a single or limited inputs, in contrast, anticipation added to a BBR system involves inclusion of

previous potential behaviors that can arise in response to multiple factor levels in the niche environment. A desired behavior choice is manifest from a range or small set of values in the niche; selection from a repertoire of preconceived responses, an anticipation set, is based on some suitable fitness to match the current conditions.

If conditions occur outside previously expected conditions, no logical or reasonable response can be made, except possibly one of total inactivity, though that also may not be adequate. Indeed, small repeated minor adjustments over time should produce desired results better than larger abrupt changes. Adding anticipation to a system should provide for adequate beneficial responses, even acting before an outcome is certain, that make it seem the system appears to know the future.

Nature and biology, indeed human existence, are filled with examples of behaviors that involve the notion of anticipation, so that when future niche conditions are right, an almost explosive action occurs to choose and execute the correct behavior to permit task achievement. In some systems, the structure develops months in advance of a process that may unfold rapidly in the future (e.g., spring flowering). Anticipation has been thought of as a trait of open systems, that includes the classification of all living systems in the world. Living systems are even able to locally reverse the law of entropy by creating organization from apparent disorder, yet at the expense of creating greater disorder, or more entropy, in the surrounding environment, and thus upholding the Second Law of Thermodynamics. Including anticipation in any system should enable tangible benefits to be realized, acting before outcomes are certain, appearing to know the future, and leading to previously determined desired results.

THESIS STATEMENT

Anticipation that is added to robot architecture should improve choice of behavior to perform desired task achievement. A congruence framework is employed to ensure that an abstracted formal system model is in agreement with some robotic natural system that has engineered desired observable behaviors. The formal system model includes equations to represent previously known archetypes of behavior, and extends them by including methods of anticipation, and thereby represents the system dynamics of operation. Simulation of an instance of a robot having anticipation is studied to gain insight into congruent behaviors expected in a natural system. The methods seek to successfully operate an instance of a real world robot, and were demonstrated to successfully manifest anticipation behavior.

Problem and Organization

Therefore, as mentioned before, anticipation has become the focus of study, and how it can be included in robotics. The intent has been to understand the entire notion of anticipation, from what it is, to how it acts, to how it is recognized, to describing a congruence framework, to defining means and methods that include anticipation in robotic architecture to effect behavior and realize task achievement. The path is an intellectual journey that recognizes and builds on existing theory, evolving into an understanding as to the role of anticipation, and culminates with use in robotics. The path starts in Chapter 2 by examining previous opinions on the notion of anticipation, and explores the traits of systems that can possess anticipation (ostensibly open systems). The task continues in Chapter 3 as a system dynamics approach is used to include known behavior archetypes. The next step in Chapter 4 develops a computer simulation model that can operate with or without anticipation, and the path eventually leads to considering methods that extend anticipation into the workings of real world robotics. The work is

summarized in Chapter 5 with statements of conclusions, and the possibilities for future work on anticipation is discussed, along with some potential related elements to pursue in more depth.

CHAPTER 2. BACKGROUND

THE NOTION OF ANTICIPATION

A behavior is a specific reaction by an agent to a condition in the environment. The behavior-based robotics (BBRs) approach directly matches a specific behavior to current environment conditions (Brooks, 1999; Connell, 1990). The behavior is a reaction that directly follows the sensed condition, and is the observed interaction between the agent, task, and environment (Nehmzow, 2000) (Fig. 1). The reaction response is a typical cause and effect relation for science that views the world as marching in one direction in time, where effect always follows cause and is dependent on a specific cause or chain of causal events (Rosen, 1991). It is so embedded in scientific thinking one hardly considers the possibility that aspects of the future might somehow affect the present, thereby inferring that time does not move in just one direction for causality.

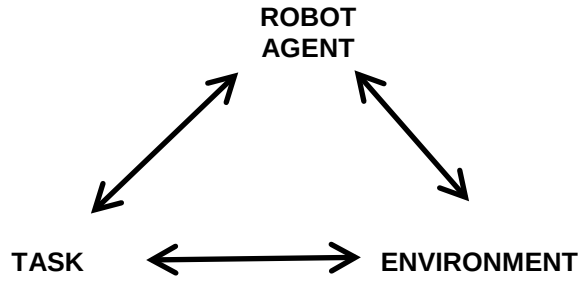


Figure 1. Task-environment-agent interlinked triangle representing behavior for robotics (Nehmzow, 2000).

Anticipation is a quality of Life that provides a means for the future to have an effect on the past (Nadin, 2002; Rosen , 1991, 2012). Anticipation involves an expectation about future events, and that expectation about the future changes behavior in the present, and also uses elements of past experience to form the expectations (Nadin, 2002). In biology, living systems can develop models of the future that can allow an agent to evaluate possible outcomes, and thus create

behavioral response to conditions that may not yet have occurred, but have a high likelihood to occur (Rosen, 2012). The likelihood is based on results drawn from past experience.

These models can be as simple as chemical pathways, chemical and physical structures, or more complex models created by thinking such as is done by humans every day (Rosen, 1991, 2012). Behavior that results from anticipation has the quality that the outcome is known *before* conditions exist in a specific niche environment to confirm that particular outcome is certain. The niche environment, or simply niche, is a specific condition of the overall surrounding environment of an agent. Hence, the nature of anticipation is that it goes beyond mere reaction to existing conditions, but looks to the future to provide for manifesting a preferred behavior choice. Anticipation is observed daily for human actions that appear to think ahead and be ready for some future event that is expected to occur. All living systems include qualities that can be termed anticipation, such as formation of flower buds that may bloom months later, or building a nest in which to rear young. Since these anticipation responses from living organisms (or living systems) involve creation of models within the organism to predict the future (Rosen, 2012), **manmade artificial systems that can create models of future conditions also possess the ability to exhibit the notion of anticipation**. Adding the notion of anticipation to a nonliving robotic agent system involves building models of the potential future world based on past experience, and using the model results to manifest the behavior choice for the agent. This requires the human designer to describe specific niche conditions that are expected to occur, and directly match a specific behavior to that niche when the niche is perceived to occur, and that observed behavior appears to respond before the outcome is known, yet it results in desired task achievement. The journey begins by enlisting a framework for abstraction and deabstraction that can instill the notion of anticipation into a robotic agent.

MODELING CONGRUENCE FRAMEWORK

An overall framework approach for modeling and prediction was described to recreate a natural system (NS) with all its entailments (implied interactions and causation) into an abstracted formal system (FS) that acts as a representative predictive model of the NS (Rosen, 2012) (Figure 2). Observations and causation from the NS are encoded into the FS using observation and measurement, allowing inferences and predictions to be made in the FS, and these predictions are decoded by prediction back to the NS for corresponding operation. For robotics, the mathematical models of theory in the FS are decoded by creation of a physical robot in the NS. Our representations of how natural biological systems operate are always model abstractions used to understand function in reality. Operation of anticipatory systems was based on a modification of this modeling abstraction by using methods in agreement with physics, and that theory of anticipation provides a basis to apply anticipation to robotics.

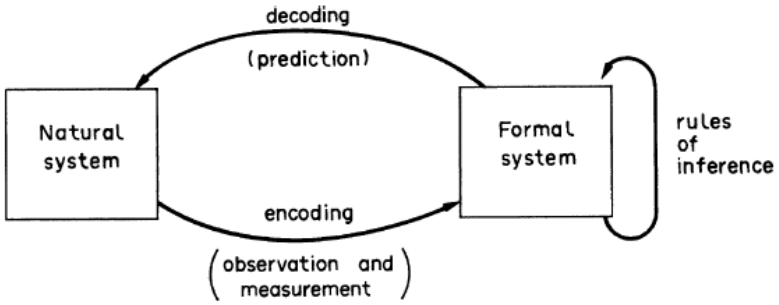


Figure 2. Representing the Natural System (NS) as a Formal System (FS), and decoding back to the NS (from: Rosen, 2012, p. 72).

A congruence framework for modeling a real world system ensures that the causality is first abstracted by encoding into a formal system (FS) of equation relations and rules, and the inference captured in the FS is decoded back to the natural system (NS) of the real world (Fig. 3). The framework is said to be congruent, or agrees, if the entailments (implications) in the

causality of the original NS can be reproduced when the inference of the FS is decoded back to the NS (Rosen, 1991). A physical engineered robot and its behaviors with implied causal entailments exist in the NS. Experimentation and measurements of observed behaviors are used to encode in a creative manner (in only one of many ways) the NS into an abstracted FS. The inferences of the entailments are structured as a model with system dynamics in an abstract architecture that a designer believes captures the desired operation of the NS. Simulations of various conditions can be done in the FS to obtain insight and ideas for application of improvements or preferred changes for the NS. Requirements and specifications are decided for the preferred operation as observed in the FS simulation, and thereby decode back to the NS, devising ways to create the behavior to achieve task performance. Various supplemental methods can be used to aid in the decoding, using scaling in both space and time, identifying key operations, developing connections, sequentially ordering the events for proper operation, attaining congruence of operations with outcomes, and devising methods of production. If the encoding, abstract modeling, and creative decoding are successful, there will be observed agreement (congruence) between the operation of the NS with the predicted performance outcomes simulated and inferred in the FS. Only with such observed congruence of the NS operation with the FS can the FS be properly called a model of the NS (Rosen, 1991).

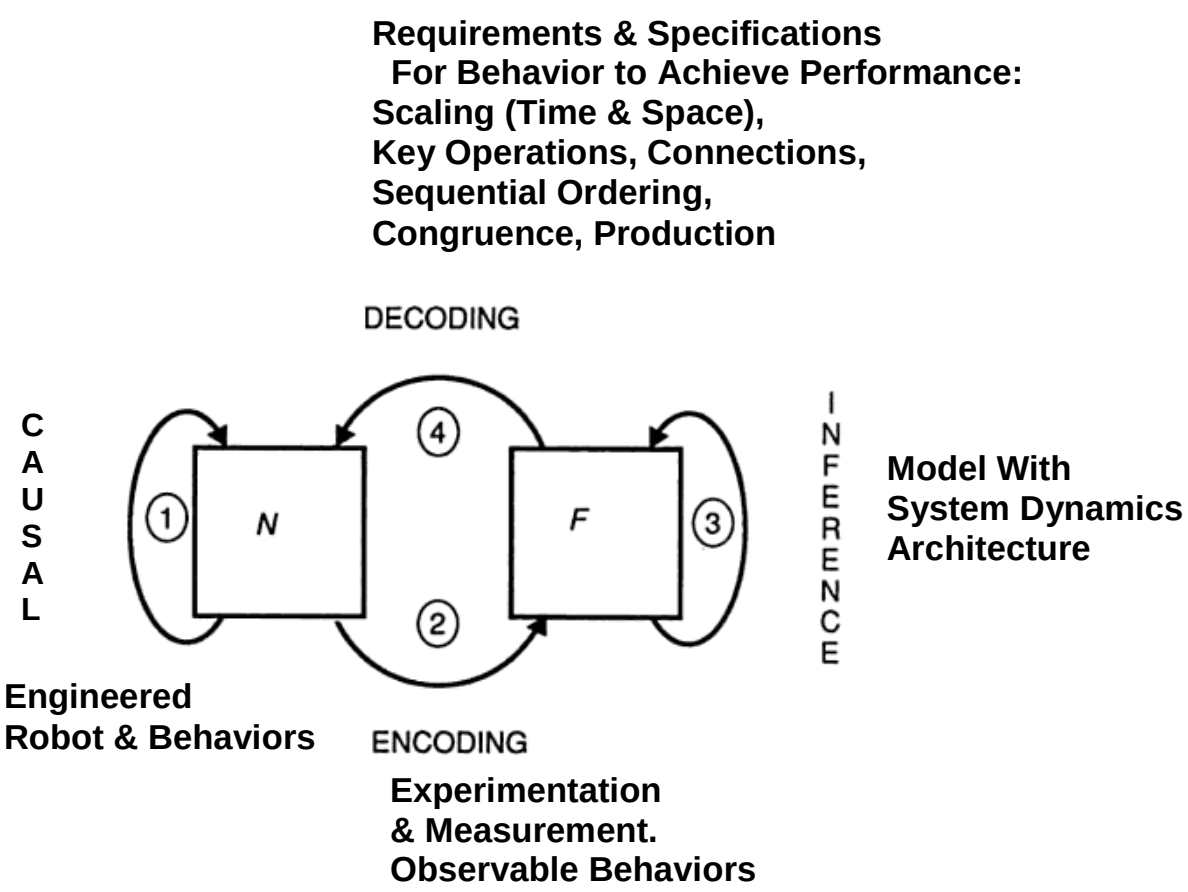


Figure 3. Rosen proposed the need for encoding and decoding to link causation relations between world phenomena into an embodied model structure.

A Natural System (N, or later referred to in this discussion as NS) can be modeled by a Formal System (F, or herein FS) by adding processes of encoding and decoding as creative acts. The circled labeled paths are related by the equivalence: 1 = 2 plus 3 plus 4, or meaning that path 1 is equivalent to the combination of the other three paths. (from: Rosen, 1991, p. 60; Fig. 3H.2)

ANTICIPATORY SYSTEMS

Rosen (2012) begins developing a Theory of Anticipation (AN) by stating the behavior of anticipatory systems (anticipatory paradigm) differs from that of the reactive system (reactive paradigm), and leads to better understanding of biological phenomena as adaptation, learning, evolution, and other basic organic behaviors. The anticipatory paradigm extends the reactive paradigm, and does not actually replace it (p. 319). Underlying principles for defining anticipatory systems are developed in a modeling context (Rosen, 2012).

Variables are assigned as S for the natural system, M for the model as a formal system, and EF for effectors linking between them (Figure 4).

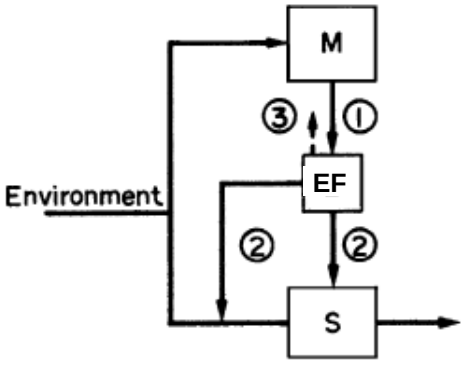


Figure 4. Interaction of the natural system, S, the formal system, M, and the effector linkages, EF, for anticipatory systems (from: Rosen, 2012).

Timing of trajectories or paths in the model, M, is much faster than in the natural system, S, so predictions of behavior generated in M are later observed in S due to coupled meaningful specific interactions. M has a set of effectors, EF, to operate on S or environmental inputs to S. Overall this is considered an adaptive system, and acts as an anticipatory system if M is a perfect model of S (or if imperfect: quasi-anticipatory). Because M is *faster*, it *predicts* the future of S. To formalize this interaction, the state space of S (and thus M) is partitioned to regions as

desirable and undesirable. If M moves into an undesirable region, the effectors, EF, activate to change dynamics of S to keep the path of S out of the undesirable region. Thus, M anticipates future activity of S, and coupling through the effectors, EF, allows predictions from M to change present S behavior to attain a chosen future behavior or situation. This creates a means by which predictions can be made about S from M, and produce an anticipatory system. A similar previous method coupled a three step delay for short term memory with both a predictor and comparator to create expectations for action from observed events (Braitenberg, 1986).

Since the theory of anticipatory systems is based in biology, a few terms from that discipline must be understood. From genetics, a genotype (or genome) is the full complement of genetic material that makes up the individual. In contrast, the phenotype is the expression of that genotype as visible in the environment. Survival and selection of the individual works on the phenotype, yet it is the DNA of the genotype that is transmitted to offspring. The biologist defines fitness as the number of progeny produced by an individual. For robotics, this definition does not apply to measure performance via behavior. Instead, robot fitness will be a calculated weighted sum of identified sensed environmental values that indicate the perceived value for performance of a certain behavior. Rosen (2012) contends that over time there are selection pressure dynamics that move the genome (the genotype) toward increasing fitness, in whatever way that is quantified (p. 343). A phenotype (or behavior) can be thought of as a path in the process, and genotype as a desired task. Selection forces organisms toward a genome having maximal fitness. This concept applies for robotic behavior selection as well: a behavior (or path) is selected to maximize fitness of performance from that behavior. How behavior is generated and fitness is assessed are independent of one another; since they involve entirely different

observables for both the environment and organism. Yet, both are coupled by some selection mechanism, and that allows phenotypes to act on genomes through the associated fitnesses. For example, behavior is generated for an organism to move towards walls or lower light locations, while fitness is determined by the structure of the organism to undertake the behavior, involving structures for perception and locomotion. The selection mechanism is effective by hiding from predators that are not able to see in the low light to catch the prey organism. Thus the behavior for moving toward low light has cues for behavior choice, and need of physical capability, but is independent of the predators' ability to catch them, which acts as the selection mechanism. The resulting evolution mechanism generates increasingly adaptive behaviors that are most fit for task achievement. Rosen (2012) represents this as a mathematical formalism that relates a desired path traveled to an actual path, and discrepancy between the paths, is defined as an area over time, A(t). The inverse, or F(t) = 1/ A(t), is a fitness observable for the two paths, and their relationship, with a larger value being more desirable as more adaptive. In this way (1) values are associated values paths, and (2) values are independent of the specific selection mechanism. There is no link between the mechanism for generating the paths, and the determination of their fitness (p. 342). A scalar field is defined on the space A of genomes, so at each point, a, there is an associated fitness, F(a). A gradient field, ∇F, is constructed on space A and from that the dynamics relation can be given as:

$$da/ds = K \nabla F(a) \qquad (2.1)$$

where K is a constant, and ds is a shorter time interval than that for the dynamics of the original time, t, for the behavior to develop. Thus, dynamics of the selection mechanism are captured in equation 2.1. Movement towards a steady state finds that value for genomes for which fitness is maximal.

Such a behavior, or phenotype, is adaptive if it is anticipatory in nature. Rosen (2012) demonstrates this by defining two related environmental cofactors, E and U. The organism can only perceive the environmental quality, E, while the adaptive behavior of the organism is, in fact, determined by the other environmental quality, U. Also, the action of an organism at the present instant has direct effect on effectiveness for later task achievement. The organism's present change of state directly effects what will happen at subsequent events or states. Since there is a link between the cofactor, E, sensed by the organism, and the related unperceived quality, U, the reaction to some value of E has a predictive relation with that of U. The relationship between the gradients of the qualities may be expressed as a function where the maxima (or minima) can be given by calculus as:

$$\phi (\nabla E, \nabla U) = 0 \qquad (2.2)$$

Determining the equation of state can relate E to U, and the associated gradients. According to equation 2.2, an organism will respond to the environment in a way to follow the desired path of U. Thus, E is treated as an indicator or predictor of U. By orienting properly with the gradient, ∇E, they automatically align with the gradient, ∇U. An important insight is that orientation with E automatically maximizes fitness at a later time. Through selection, the organism has generated a prediction about how present behavior will affect future task achievement. Rosen (2012) cites an example, where E is a measure of light (that can be perceived and used by a phototropism), while U is a measure of predator density. By following the negative gradient of light (towards darkness), the organism automatically follows a negative gradient for predator density, and thus avoids predators. It maximizes fitness by moving towards dark now (behavior), and results in an opportunity to live to reproduce later (task achievement). Interestingly, the actual mechanism for the behavior, that of moving in the gradient field, is independent of the

relation between the gradients. The notion of fitness can be considered a common currency for choosing between events or behaviors, similar to energy in physics or value in economics. The application of the notion of fitness to robotics should enable a means for choice of behavior based on anticipation about future events.

Rosen (2012, p. 320) illustrates the notion of anticipation with a biosynthetic pathway that acts as a simple anticipatory system (Figure 5). The concentration of precursor substrate, P0(t) at time, t, affords forward activation (Anticipation) to control activity rate of catalyst, Cn, that controls the rate of conversion of Pn-1 to Pn at a later time, t+h. The system is anticipatory since P0(t) is a predictor of the later concentration and

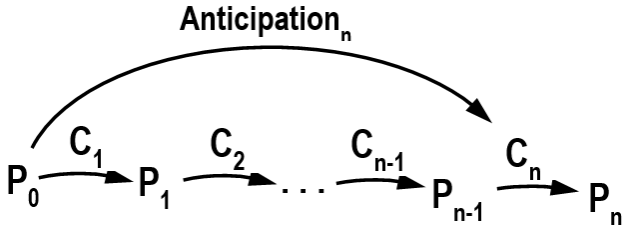


Figure 5. A metabolic pathway shows the notion of anticipation by a predictive model for Cn to catalyze Pn-1 at a later time, t+h (modified from: Rosen, 2012, p. 320).

reaction of Pn-1(t+h) with activated Cn at the later time, t+h. By modulating Cn, P0 pre-adapts the catalyst to process the substrate Pn-1 at a future time. Balance (or homeostasis) is maintained only through the predictive modeling relation between initial P0 and later Pn-1, and that relation links the model prediction to the rate of catalyst enzyme action of Cn. There is no feedback in the pathway, and no mechanism to measure the quantity that is actually controlled (p. 323). Since increase in Po converts a greater amount of Pn-1 to Pn by catalyst Cn, system dynamics identifies this as a reinforcing loop, though with no feedback, that continually is

unstable as an ever increasing trend. In reality, some system dynamics balancing loop will counteract and control the whole process by some limiting factor to constrain amounts to some asymptotic level. The constraint may be as simple as some means that restricts the amount of Pn that can be produced, possibly by limiting precursor substances for the reactions.

ANTICIPATION PRINCIPLES

A comprehensive theory of anticipation (AN) is considered difficult by Nadin (2002, p. 53) because awareness of AN is not easy to attain. He presents AN as feedforward thinking that will create determination in a future state, and that produces feedback causation to the present state (p. 53). He offers several definitions relating to AN.

1. An anticipatory system (AS) has a current state determined by a future state.
2. The source of AN is interaction between minds, and shared experiences.
3. An AS has predictive models of itself and/or environment, allowing instant change of state.
4. AN arises from a correlation process, thus allowing an organism to anticipate sensory data, or act on scarce data.
5. AN is an expression of connectedness with the world.
6. AN is a mechanism of synchronization and integration, and an attractor in dynamic systems.
7. AN is a recursive process. To an external observer, the system appears to act as if it knows its own future.
8. AN is one realization of an instance of many possibilities. AN is a realization of a possibility.

AN is shown in many areas in the world: building a cyclotron to find neutrinos (p. 83), migration by birds, birds increasing song production at the start of breeding season (p. 73), growing longer fur by mammals before winter, trees preparing to lose leaves for winter based on daylength,

genetic resistance to specific diseases, investment decisions, swarm behavior by flocks (p. 78), playing of an orchestra (p. 78), and adaptive robotics.

SYSTEM TYPES

General Systems Theory

A field of science, termed General System Theory (GST), as developed by von Bertanffy (1968, p. 253) strives to formulate and derive general principles that apply to any systems in general over all of science: wholism, differentiation, centralization (or the convex), finality, causality, and isomorphism. The previous view of a Newtonian reductionist mechanistic world viewed living beings (and humans) as machines, but is replaced by one of whole systems with vital embedded relations that promote understanding of even life itself (von Bertanlanffy, 1968; Rosen, 1991). The whole of human culture and society depends on structure dependent on language and human derived symbols (von Bertalanffy, 1968, p. 251 & 252). Two aspects arise from this: 1) specificity of human history, shown by traditions (in contrast to heredity of evolution), and 2) mental experimentation using conceptual symbols to achieve goal-directedness for production and reproduction of life in a whole organism (permitting the organism anticipation of an expected future). The teleology of these aspects are explained by von Bertanffy (1968, p. 252):

> "True purposiveness, however, implies that actions are carried out with knowledge of their goal, of their future final results: the conception of the future goal does already exist and influences present actions. This applies to primitive actions of everyday life as well as to the highest achievements of the human intellect in science and technology… [I]t is up to him [read: mankind], however, whether he applies his power of foresight [read: anticipation] for his enhancement or his own annihilation."

Thus, the language and symbolism within a human mind creates the organizational structure with potential not only to see and perceive a specific future, but to anticipate a path to that goal that

invokes action and behavior *before* an outcome is certain to lead to the desired result. Mental experimentation can construct a new world before it exists in physical reality. As a result, *the future influences the past*. A general aim of GST is to find isomophisms, or a one-to-one correspondence mapping between objects in vastly different systems in different fields (http://www.politicalsciencenotes.com/articles/general-systems-theory-concepts-and-limits/510, von Bertanlanffy, 1968, p. 80). Scientific laws can be thought of as abstractions or idealizations to express aspects of reality, such as ensuring a design on paper corresponds to (is congruent with) some construction in the physical real world (von Bertanlanffy, 1968, p. 83). In other words, science shows perceived orderly traits of reality have conceptual constructs. Implied is that order exists in some reality. A system is viewed as a number of interacting elements with relations expressed as differential equations of the form (p. 56 & 83):

$$dQi/dt = fi\,(Q1, Q2, \ldots Qn), \quad i= 1, \ldots, n \qquad (2.3)$$

Reworded: the changes per unit time for each significant element is a unique function of the current values for all of those elements. This implies a type of entailment, causality, and finality (as found in the Rosen congruence framework) where a set of element values implies the specific changes that ensue. Three descriptive levels can be expressed in concepts of relation: 1) analogy as superficial similarities between relations with no correspondence as to cause or relevant laws, 2) homologies having different effective factors with the same formal underlying rules, and 3) explanation that provides a set of conditions or rules valid of an individual item or class of items. By comparison, analogies are not useful in science, while homologies can present useful models, not necessarily reductionist in nature, and with a formal correspondence between abstraction and reality for various kinds of systems (p. 85). Additionally, explanation replaces the general

differential equation form with *specific* differential equations for a *specific* individual case, with direct correspondence between the equation rules and the individual case.

Open Systems Enable Self-Organization

For a system to be self-organizing, as for living organisms, von Bertalanffy (1968, p. 96) discusses that Ashby has shown that a self-organizing system must be an open system (OS), one that exchanges flows of resources (in and out) with the environment. Though Ashby starts with a dynamic system of differential equations to describe the machine homology, von Bertalanffy counters this as too restrictive for biological systems rampant with discontinuities (p. 96). Ashby explains the self-organizing system maps inputs into the next state of the system, in one of two ways: 1) separate parts of a system form connections, or 2) the system changes from a nonoperational to an operational one. But Ashby states no system as a machine can do this, since the change does *not* occur from *inside* the system. Instead, some *outside* agent provides input that changes the system. Thus, for any machine to be self-organizing, it must couple to something outside that system (p. 97), so the system machine is not closed, but should be termed open. Input is required for self-organization of any system. Organization implies decreasing entropy, by definition. Recall from the Second Law of Thermodynamics that order is ever decreasing for systems, so entropy (which is the degree of disorder) is always increasing over time (Rifkin, 1980). To conform to the Second Law of Thermodynamics, disorder must occur somewhere, and for the self-organizing system that is actually outside the defined system. So entropy decreases inside the self-organizing system, while the inputs and outputs of the system transfer increased entropy to the external environment (p. 97). Energetically, organization can increase inside the living biological system, but the inputs (e.g., oxygen and nutrition) make this

possible, while outputs (e.g., CO_2 and waste products) pass increased entropy or disorder outside the system to the environment (p. 98). Obviously, any organization, including self-organization, has a cost in terms of entropy increase to the environment. Robotic systems having anticipation are actually open systems that still follow the entropy law to create disorder in the environment. Thus, all behavior will lead to some larger form of disorder even when the robotic agent appears to create order, if only by use of energy from the environment, and some type of waste generation such as by structural degradation and need for maintenance or replacement parts.

Whole Systems

Health for both humans and robots is considered relevant (Goldstein, 1995, p.11, first printed in 1934, in German). Well-being allows for ordered behavior in spite of limits (p. 11) imposed on the organism. Symptoms are attempted solutions undertaken by the organism, and may be either successful or unsuccessful. In the forward to "The Organism: A Holistic Approach" by Goldstein (1995), a theme is discussed by Oliver Sacks (p. 29) concerning pathology and its value to the nature of health. The notion of order is central to health:

> "Thus, being well means to be capable of ordered behavior. which may prevail in spite of the impossibility of certain performances which were formerly possible… Recovery is a newly achieved state of ordered functioning [as] a new individual norm."

Symptoms are both an attempted solution, and an adaptation to an altered inner state (and world). A Holistic approach seeks to understand behavior of the organism as a whole. A biological approach deals with brain damage (from war), and leads to a theory of understanding organism function (Goldstein, 1995, Preface, p. 15). For robotic agents, maintaining correct operation of the agent for task achievement is needed for the robot to be considered healthy.

In his introduction (p. 13), Goldstein contends that using an approach lower to higher is incorrect, such as to study lower organisms (e.g., birds) to understand higher ones (e.g., humans). Instead, to study humans directly allows one to understand the whole organism. Organisms are made simple only through abstraction, and that is a misplaced concept, since all living organisms are in many ways quite complex. By the study of simpler organisms: we simplify them artificially (p. 14), which is an inappropriate approach. The nature of simpler beings is so remote from us that we lack any real understanding of their functional operation, and the potential complexities they possess. Gross mistakes are best avoided by studying human behavior directly [or for any other such complex agent, such as a robot]. Goldstein opposes transferring findings from one field/being to another, and also thinks it is wrong to apply human findings to animals. Thus, robotic agents should be studied directly in their niche context to ensure the resulting behavior in actually correct and congruent for desired task achievement. However, he agrees that study of the central nervous system (CNS) may generalize to other organisms with similar systems of a CNS. Goldstein (1995) describes his overall view for formalization of such a process as:

> "Any formalization of the subject matter of a science is useful only if it follows, not precedes, the investigation. This inevitably must be the case since the subject matter itself becomes apparent only during the process of research, as it emerges from the indefinite province in which it was embedded. This is equally true for biological research."

This stance takes the approach to first study an organism through observation and experimentation before developing any theory as to the operation of such an organism or system. Goldstein's view tends to conflict with the observation of Rosen (1990) that contends the use of metaphor has been used, though not as true science, to study and predict about systems before a true encoding of causality is conducted from the natural system to inferences in a formal system.

Yet, Rosen contends rightly the use of metaphor limits the possibility to verify the proper application of a formal system abstraction to the desired real world natural system.

Motivation For Organisms

Maslow's theory of human motivation identifies five prioritized levels of needs that determine behavior, from physiological (highest priority), to safety, to love, self-esteem, and lastly self-actualization (lowest priority). An average person actually is motivated by partial satisfaction or gratification in all these categories, requiring higher satisfaction in the higher priority categories. By homology of structure (more comparable than analogy) for motivation, a robot also has highest priority to maintain its physical structure (akin to a physiology). The human designer should focus most strongly on building a robust lasting structure that operates in its world environmental that is not endangering the physical structure or programmed operational behavior. Robot behavior should always seek to preserve the basic physical structure, and thus potential for operation. In the next lowest priority category, the human designer must include physical structure and behavior choice that ensures safety of the robot, human beings, and nearby surroundings key to continued desired operation. Behavior of the robot would appear to be motivated by safety, or as Maslow puts it for humans: "…we may then fairly describe the whole organism as a safety-seeking mechanism." (1943b, p. 376).

It would seem more difficult to ascribe the lower priority categories to a robot agent (love, self-esteem, and self-actualization), since we think of them as purely human qualities. Yet, Maslow also states the 'love' category as a 'belongingness' need (p. 380), in which case the robot and processes belong to groups and networks or even model types that the behavior of the robot

could be accurately labeled as belonging to provide for task achievement. The ‘self-esteem’ category also is identified to be ‘soundly based upon real capacity, [task] achievement, and respect from others’ (p. 381). Hence, robot capacity for performance, task achievement as core to behavior-based robotics, and respect (or appreciation) of ability to perform by human observers or control mechanisms endow a type of esteem onto the robot agent, notably from an outside source or observer. The ‘self-actualization’ category is even more challenging a homology. Though the statement by Maslow: ‘What a man can be, he must be.’ (italics in original, p. 381) could find a parallel in: ‘What an agent can be, it must be.’ Here the parallel statement implies a purpose for which the agent is designed and structured, and obviously the agent must perform that task achievement, as would be observed and measured by someone else or a control system. Curiously, Maslow points out that in society, people that are basically satisfied are the *exception* (italics added here), not the norm, and thus at the time of his proposing of the theory there was little experimental or clinical evidence for self-actualization, with the need for further research (p. 383). With little evidence available even for humans, the proper application of self-actualization to robot agents also requires copious additional research and homology of principles. Recall in all cases for categories, Maslow proposes the theory for motivation of a behavior. From such a point of view, a robot agent is observed to make a behavior choice according to an internalized motivation that overall fits Maslow’s theory for motivation, so it may apply to robots as well as humans, using the homologies discussed immediately above.

More broadly applicable is the assertion by Maslow that motivation of a behavior is based on multiple basic needs simultaneously (behavior is multi-motivated, p. 390), and that there are other multiple determinants aside from basic needs and desires that determine the behavior

choice (p. 387). Similarly, robot agent behavior is surely based on multiple cues that combine to create the behavior choice. Maslow (1943b, p. 389) suggested that unconscious needs and associated motivations are more important than conscious ones, and since we as humans do not ascribe consciousness to robot actions, the driving force for robots can easily be said to be unconscious or subconscious. An attempt was described by Malcolm and Smithers (1990) to structure robot processes in an architecture of subcognitive behavior modules that encapsulated the cues and specifics of behavior, while a driving force was derived from cognitive modules without the details of behavior. This approach implies a type of cognitive consciousness identified as the observed active direction for choice of behavior. In addition to motivations, as Maslow (1943b) contends, the choice of behavior depends on the field or context the agent is situated in, as well as external stimuli (sensed cues), association of ideas (or rules), and basic reflexes (e.g., basic motions possible). The two main types of behavior and combinations of them can be distinguished as expressiveness behavior (something a robot might do for show as entertainment and play, or display of built-in capability) or more coping behavior that is purposeful in attaining a goal of task achievement (p. 391).

Anticipation is alluded to by Maslow in two ways. First, he refers to the work of Goldstein (1995, reprinted from 1934 in German) that whole organisms *avoid the unfamiliar* by attempting to maintain orderly surroundings, so that unexpected dangers cannot occur (p. 380). An unexpected event is labeled as a grave danger, causing a panic reaction for humans. If panic reaction can be observed as an unexpected or illogical reaction to the outside world, then a robot agent may react in a parallel manner with *unsuccessful behavior to some unexpected event*. Since there is no anticipation of the event in the robot repertoire, no logical or successful behavior

results. Only situations or events anticipated by the robot (included in its structure and programming) can evoke successful behavior that results in task achievement. Otherwise, a seemingly 'panicked' illogical unpreferred behavior occurs. Second, Maslow also illustrates that both current and future world outlook or philosophy for humans is dictated by *how well basic needs across categories have been met in the past* (p. 376). If a human appears solely as employing a safety-seeking mechanism, it infers all safety needs have not been satisfied in the past, thus creating anticipation that the safety needs must be the focus to be met in the future. The anticipated future includes a philosophy of the future that all else is less important than safety, where even physical or physiological needs have been satisfied and are now underestimated. Here the situations of the past have created the expectations for what is anticipated to be focused on in the future. The focus for the future is on what is most anticipated to happen. Thus, for robotics, situations encountered previously that have a *high expectation to be met in the future* create anticipation by the robot agent to make a choice behavior from an existing repertoire of behaviors for desired task achievement. The goal of the robot designer is to include a choice of behaviors to cover anticipated past events that are expected to occur again in the future, thus assuring task achievement.

Behaviorism as Studies of Unconscious Actions

Watson (1913), referred to as the father of behaviorism, discussed how a behaviorist should view the field of psychology as a strictly experimental branch of science intent on predicting and controlling behavior (ostensibly in a positive manner). He scoffed at introspection and using consciousness to interpret meaning and results. He reasoned humans will respond biologically similar to other (closely related) animals, so experiments on animals can be deductively applied to humans, an idea that conflicts with the specific whole organism approach of Goldstein (1995,

reprinted from 1934 in German). Focus should be on the external observable behavior and reactions of people rather than their internal mental state. Words and memory are devices to permit thinking, or as Senge (2006) stipulated that allows thinking as a process that leads to thoughts, and these in turn lead to action or behavior. For Watson, with emphasis on behavior, emotions are only incidental physical responses. Since studies of consciousness for causation resulted in various different and inconclusive outcomes, conscious should not be considered as an element of scientific psychological study. Instead of a study of the mind, psychology should study behavior of individuals, and not individual consciousness (from: https://en.wikipedia.org/wiki/John_B._Watson). As a homology with robotics, implications of consciousness should not be considered or studied, only the conditions that resulted in behavior.

Simple Systems

Design as Choices

Decision making and the theory of design has been presented in a rigorous manner in a curriculum that defines present commitments of resources to follow choices toward desired outcomes (Simon, 1996). Such decisions are made with *bounded rationality*, meaning not all information is known that affects the outcome of the decision, and realize that future contingencies that do not affect the present commitments are irrelevant to the design process (Simon, 1996). Simon discussed the evolution of complex systems from more simple ones, most likely by combining stable subassemblies in a hierarchical manner. An agent performs a task using simple procedures in a complex task environment (TE). Complex systems, defined as those with many interacting parts, most likely arise from stable subsystem blocks, while previous experience selects the most successful path to a desired outcome. Methods to maintain

homeostasis and provide correcting feedback lead to adaptation and coping for the future. Though rigorous methods for optimization might be used, in most cases a *satisficing* solution is discovered using bounded rationality that is acceptable but not the absolute optimum, since too much time and too many resources would be needed to search across an almost infinite number of possible solutions. Also, Simon (1996) suggests it is most important to define a problem into the proper representation to see a successful transparent solution.

A tale by Keijzer (2001, p. 106) illustrates that a robot may spend so much time divining between relevant tasks and an infinite number of irrelevant tasks is stuck in the process of deciding (or indecision), and never is able to act. This may be termed the *frame problem*, where considering too many irrelevant alternatives results in inaction and task incompletion. For any uncertain task, one might find a very high number of alternatives that could not be performed by the agent. Thus, the agent must employ a means of deciding a path based on constraints, and a designer is required to initially set as many such constraints as possible. This also defines what tasks, or environments, are considered unsuited for the agent behavior.

Journey of the Ant

Science unmasks complexity to describe with simplicity. What scientists observe may differ from actual underlying mechanisms in the NS (Simon, 1996). For an autonomous robot, the goal is to turn on a switch and observe that it works for task achievement. This is similar to an ant that moves in its niche on a sandy beach. In an illustration that is known as Simon's ant, simple movements by an ant are able to traverse a complex niche. Realize here the niche environment is the complex thing, not the ant. (http://everything2.com/title/Simon%2527s+ant and Simon,

1996). Principles of engineering should be used in a simple direct manner to design and build robust and reliable robots in a specific real world niche. The goal: design and synthesize a specific solution for a specific real world concept or problem.

Future Course

As discussed, Simon (1996) has contended the complexity of the path of an ant is in the context of the niche environment, not in the operational rules of the ant itself, so understanding the niche is as important as understanding the inner workings of the ant locomotion. Interestingly, in the history of science all past theories have turned out to be false, so those we hold to now probably will be false as well (Dupre, 1993). In the field of robotics, this already has transpired as the sense-plan-act horizontal architecture of classical robotics (CR) has been shown to be replaced at least in part by the sense-act vertical architecture of behavioral-based robotics (BBR) in a subsumption architecture that creates levels of behavior through module interactions that result in robots that are situated and embodied as physical real world robots (Brooks, 1999). To return to the thinking of Simon (1996, p. 128), evolution is a design process that devises multiple alternatives, and then selects the best agents through multiple tests on a nested series of cycles. Humans solve problems by a mixture of trial and error using some means of selectivity heuristics (rules of thumb) in the simplest process possible (p, 195). Selectivity is aided by relying on stable blocks for subsystem configurations to build up from incremental stable blocks into more complex configurations. Previous experience can be used to identify a previous viable solution, and build on that further. Thus, both stability and experience are needed, or at least preferred, to perform successful designs even in natural evolutionary settings. Simon goes as far as to describe genetic DNA as software (p. 213), so that state and process are entailed in DNA. DNA constructs

and maintains the organism, contains the recipe for programmed instructions, and makes copies of itself and subassemblies. Ontogeny recapitulates phylogeny, where developmental stages are rebuilt from the past, so that many problems are reduced to ones previously solved (e.g., an embryo becomes a fetus, etc.). Overall, a whole system is constructed of many interacting parts in a hierarchy of 'nested' subsystems (p. 184). The concept of *satisficing* is always present, so that a workable solution is found, though it may not be an optimized or theoretical 'best' solution, at least due to time constraints (p. 119)

Churchland's Crab

A simple 'crab-like' fictitious creature can be constructed to have a sensory grid as a point by point projection from an orthogonal grid at the eye receptor to a 'distorted' view in motor space (Churchland, 1986) (Fig.6). A mapping function, $(\theta,\psi)= f(\alpha,\beta)$, maps the motor arm angles (θ,ψ) as a function of eye angles., (α,β). Coordinates in real 2D space (a,b) for the eye angles are:

$$a= -4 (\tan \alpha + \tan \beta) / (\tan \alpha - \tan \beta)$$

$$b= -8 (\tan \alpha * \tan \beta) / (\tan \alpha - \tan \beta)$$

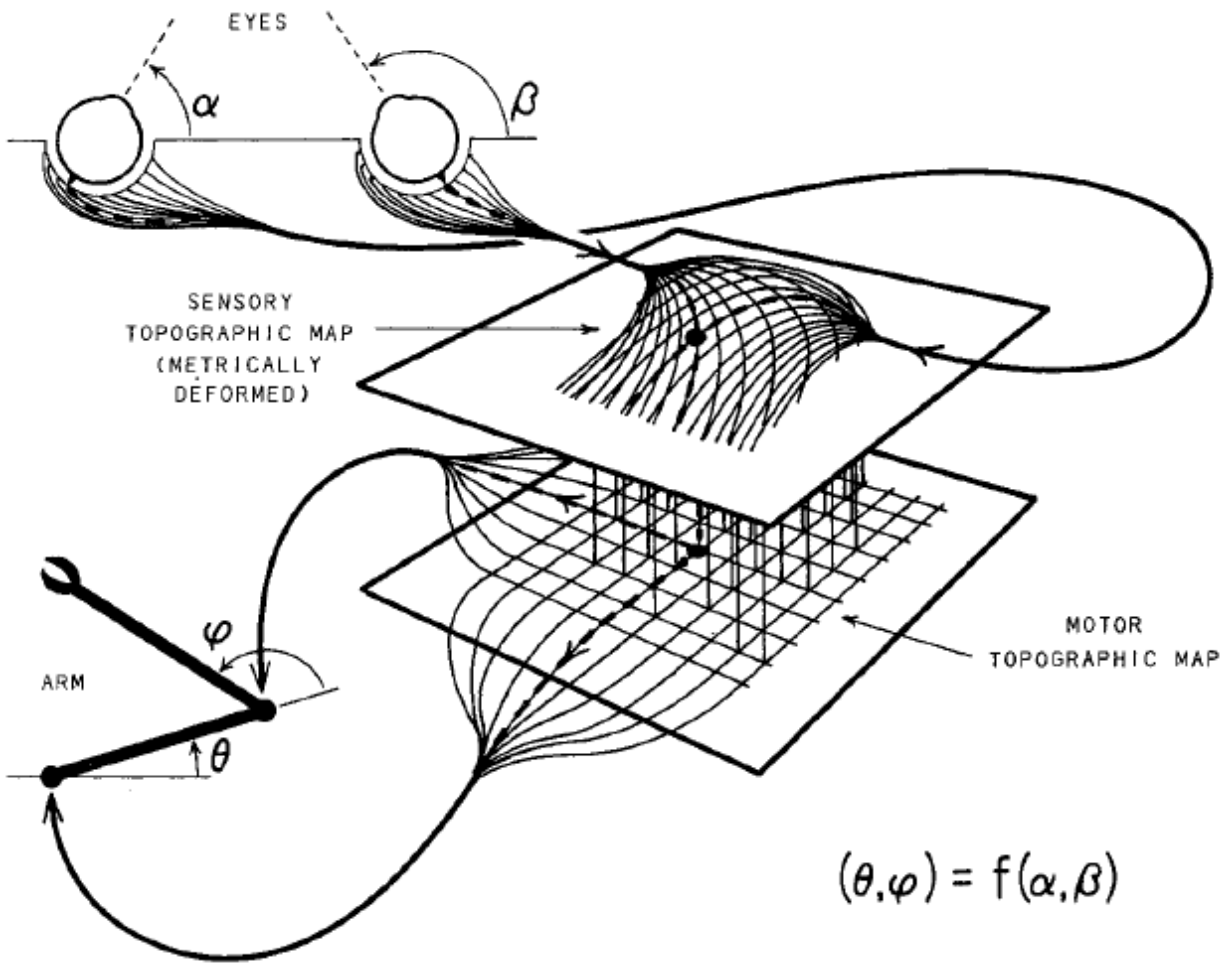


Figure 6. Transformation mapping of the 'crab-like' schematic of a fictitious creature from orthogonal sensory eye space angles (α,β) to 'distorted' nonlinear motor arm angle state space (θ,ψ) [from: Churchland, 1986, Fig. 6].

More elaborate equations can relate (a,b) to elbow coordinate angles. Deformations are mapped to an appropriate correspondence, termed the *state-space sandwich*. Damage to a small portion of the sandwich still allows for operation. Indeed, the system is considered to be fast enough with biological conduction fibers to conduct the movement in real time. Nonlinearity of the mapping infers a small error may be magnified disproportionally, which may create undesireable large errors. This structure is biologically realistic, and could be grown by life. Overall, it is an

example of a simple connected biological system that translates eye movement into movement of the crab claw to reach the desired item that is being looked at.

The goal for minimalist robotics is to construct such simple mapping relations using physical connections from one point to another so little or no processing is needed in between. Of course, such connections and changes can be made with switching within programming and processing of a microprocessor, having direct connections made from perceiving sensed inputs to the corresponding desired activated outputs. Therefore, the preferred *current* method for application of such a mapped system is to incorporate *both* robust physical connections along with simply activated internal electrical microprocessor connections that can directly lead from sensing to acuation. The desired goal is to avoid complicated calculation and analysis that delays processing time and overly complicates the decision for performance of the needed action.

ANTICIPATION PROMOTES HABITS

The nature of habits in organisms is explored by Duhigg (2014). He cites an example how neuron excitation in a rat's brain decreases with repeated runs through a familiar unchanging maze to reach chocolate (p. 16) (these rats were upgraded from cheese…). The perceived change (a form of learning) in brain activity reflects the formation of a habit. Duhigg contends habits form through the sequence of a cue, routine, and reward. The routine itself is formed by a process termed chunking. [In psychology, chunking is combining several items together in a way so they are remembered together and easily recalled later. Letters in a word are a form of chunking, and repeated expressions are chunking of smaller chunks.] Sound cues at the start, as clicks versus meows, may be used to initiate a specific behavior that easily reaches the chocolate

reward. The cycle of cue, routine, reward is defined by Duhigg to be a reoccurring habit that a person, or any organism, can be fixed into performing repeatedly.

Habituation is defined in biology as a form of neural plasticity in organisms that is a simple type of learning (BZ580 Biological Basis of Behavior notes, CSU, 2010-9-21; https://en.wikipedia.org/wiki/Habituation). Neural plasticity reflects a physical and/or chemical change in an organism's neurons that cause a change in behavior in the future. Habituation is a decrease in response to repeated stimulus presentation (BZ580 notes). For example, one stops smelling bacon or fish smell in the house after a bit of exposure to that stimulus, yet you still smell other odors. After one leaves the house and returns, dishabuituation occurs so that one can smell the bacon or fish again. A classic example of habituation is the sea slug (*Aplysia californica*) that is poked and it withdraws its siphon initially. With repeated stimulus poking applications, the duration of withdrawal decreases (Fig. 7). Habituation may continue longer term depending on the training (practice) regime. Such studies show the nature of altered neural impulses that reflect a change in physical neuron firing and the resulting changed behavior, similar to the example in rats given by Duhigg (2014). The mechanism is reduced neuron firing after repeated training by stimulus application.

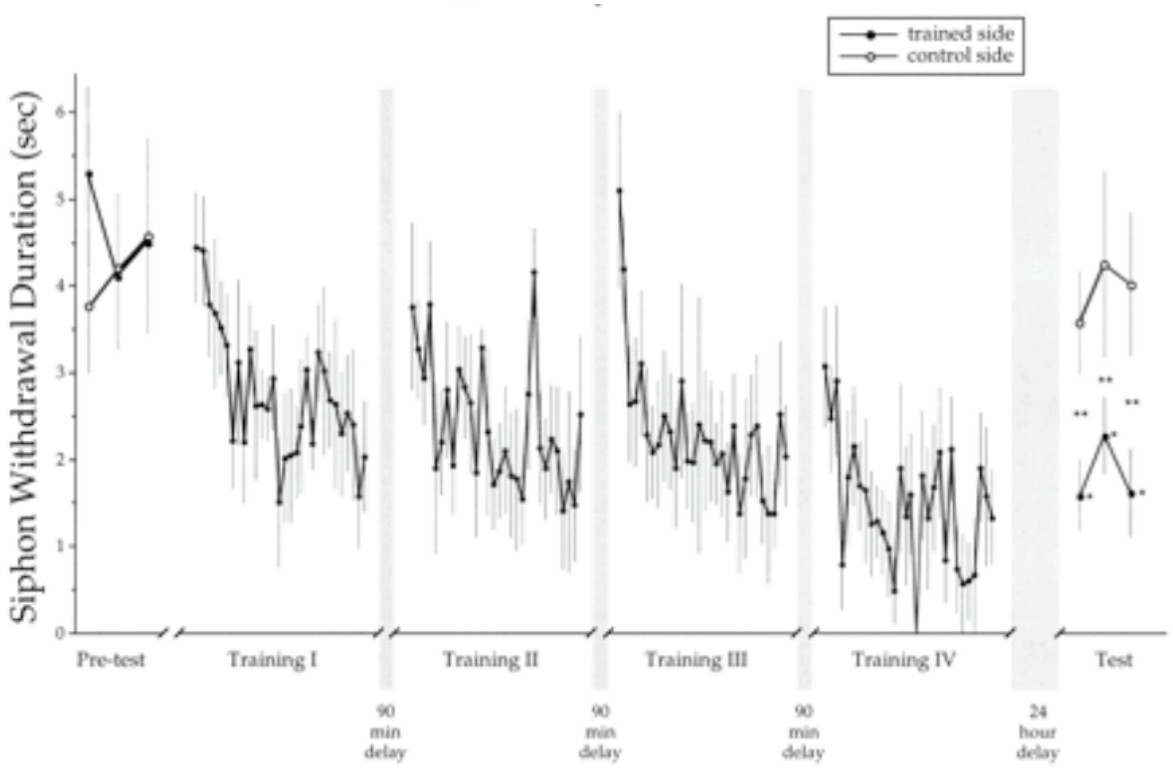


Figure 7. After repeated training times of poking the sea slug, habituation occurs over time so that the duration of siphon withdrawal is reduced in response to the poke stimulus.

Fixed action patterns (FAPs) are similar habitual responses to cues that are defined by behavioral biologists (Tinbergen, 1951). Examples in biology range from a goose using a scooping notion of its head and neck to return an egg into the nest, to a bird feeding fish in a pool because the open mouths of the fish resemble those of young nestlings, and thereby cues the feeding action response by the bird. Humans use similar cues to initiate getting up in the morning with an alarm clock, or turning at a specific corner while driving to reach a familiar destination. In that sense, much of human daily behavior can be thought of as automatic and unconscious, and may occur similarly for other organisms as well.

Anticipation is an accessory to habits. Duhigg (2014, p. 46) explains how a monkey forms a habit by reacting to a shape displayed on a computer screen: it presses a lever to obtain a raspberry juice reward. Over time, the electrode in the monkey brain changes profile response.

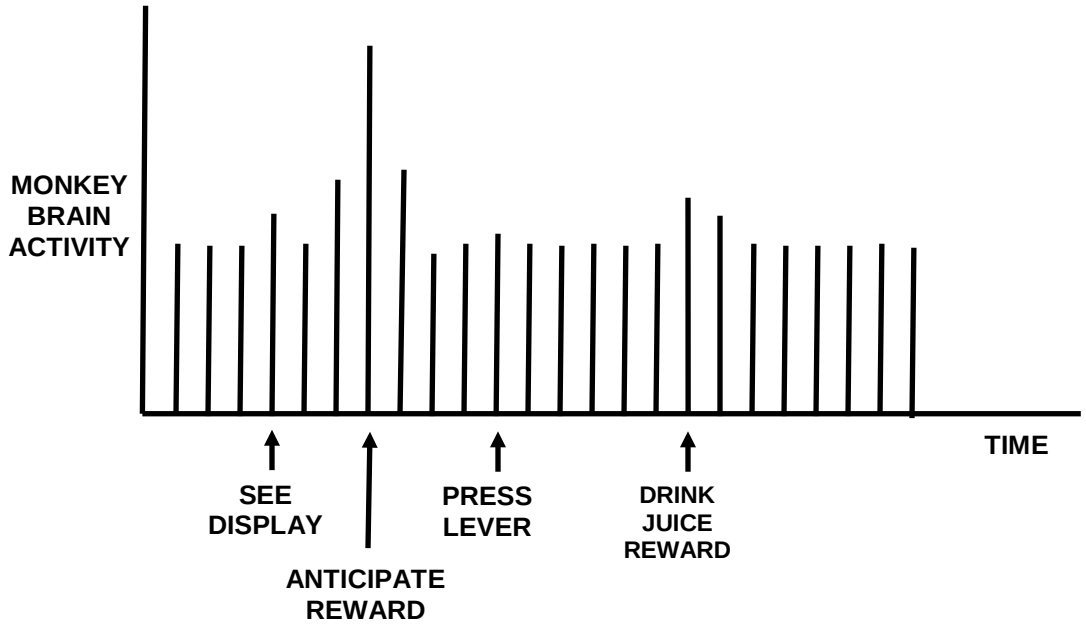


Figure 8. After repeated training times of providing raspberry juice to the monkey, anticipation of the reward increases prior to the lever press and juice reward, and drink juice reward response is reduced [modified from: Duhigg (2014, p. 46)].

Initially, the neural response spike is highest just after receiving the raspberry reward. After some time, the spike in response occurs just after seeing the shape appear on screen. This is *before* the pressing of the lever and the actual reward (Fig. 8). This seemingly early response is an indication of anticipation of reward by the monkey brain. The combination of forming the habit and anticipation of reward is termed a craving (expecting the reward to come). An intersecting response behavior becomes associated with this. If the reward continues as anticipated, all is well, and we have a happy monkey. However, if the reward is delayed or

removed, we now have a very unhappy monkey that shows signs of anger and depression. The moral becomes that if we no longer receive the anticipated reward, we become irritated in the execution of the habit, although we continue to apply the routine response.

Changing a habit actually involves linking the cue to a new routine, and preserving some reliable reward after the routine completes. Duhigg (2014) explains that the benefit of recognizing the habit (cue, routine, and reward) is that one is then able to replace the routine after the cue, and thereby convert an adverse addiction habit into some more favorable desirable habit routine. In a more complex application, a National Football League (NFL) coach of the Tampa Bay Buccaneers, Tony Dungy, attempted to change and redirect his players' behavior by instructing them to focus on small specific details as cues. He had the players practice repeated specific routine responses to the small specific cues until the response was so automatic that the players no longer had to think (p. 64). By *not* requiring them to think, the response was more direct and faster to the initial cues. That extra speed in decreased response time, although only milliseconds, was enough to provide an edge so that his team outperformed the opponents, even though the opponents might know what was coming. The outperformance leads to scoring and winning as a reward. Hence, in this example thinking actually got in the way of execution, by delaying reaction time, and was not preferred. Ingraining the habit response to the cues was more important. Indeed, the focus here was to `develop direct input perception to output connections similar to those operating for Chruchland's crab (1986). Little processing was needed, so response was faster, and in sports competition that is the preferred edge that one would expect to lead to more consistent winning, whatever the sport.

In parallel, though Duhigg did not explicitly discuss it here, if the cues provide an anticipation of the reward, the behavior can be encouraged to occur more expediently, and in this case is more desirable to produce the task achievement outcome. The lesson here is that proper practice to a small number of cues can set the desired habit response that performs more quickly and effectively. Anticipation, as shown earlier with the monkey (p. 46), unconsciously becomes included in the process, after the cue and before the actual routine, that is preemptive and enhances the routine action that attains a desired craved reward. Anticipation becomes an unconscious prior-occurring supplement to the habit routine that enhances its performance. Thus, anticipation is beneficial as a performance enhancer.

Indeed, for any general system, defining of a set of desired behaviors constructs the types of 'habits' that are expected to be observed in the specific environment being considered. Proper definition of these behaviors should devise an anticipation set of possible behavior selections. Specific cues that are perceived in the immediately surrounding niche environment should directly manifest the desired behavior for that condition, or group of conditions. Just as a monkey brain can be 'wired' to perform the habit (cue, routine, and reward), a robotic system can perform behaviors that appear to be 'habit' due to internal wiring of the connected components and actuators. Also like the monkey brain, the cued habit can be enhanced to perform more quickly and efficiently by including predetermined connecttions for direct linking of the cue to the desired performed action. This predetermined expectation of the performance can be thought of as anticipation that can be built into any automated system. Therefore, anticipation can be added to robotics by incorporating predetermined routines that enhances performance of a behavior that appears to act like a habit.

ARCHETYPES

Discussion Versus Dialog

The concept of business archetypes was proposed in relation to team cooperation and learning by Senge (2006). Although the idea of a generalized archetype was proposed earlier by psychologist Carl Young, showing archetypes as broad patterns of ideas (e.g., beauty, nature, or God), Senge applied them to the business world to describe recurring problems and develop solutions. Solving problems requires exchange of ideas, leading Senge to discuss two types of discourse: discussion and dialogue (p. 223). Discussion has a goal to win a point versus finding truth and coherence. In contrast, dialogue is a free flow of meaning between people, and goes beyond understanding of one individual. It gains insight that cannot be gained individually. The purpose of dialogue is to reveal incoherence in thought. In dialogue, people observe their own thinking. People together observe the collective nature of thought. Thus, thinking is a process; thoughts are the result of the process. People tend to ignore the thinking process, and instead tend to own thoughts. Yet, collective learning arises, and is vital to potential for human intelligence. Dialogue shows incoherence in thoughts, allowing more coherence of collective thought.

Team Learning

Dialogue is important because it is ‘open to a flow of larger intelligence’, and has goals of truth and coherence. However, discussion (p.223) has a goal to win a point versus truth and coherence. It is presented that all scientific theories are eventually proved false (p. 222) (quote Bohm, 1965, The Special Theory of Relativity.), as was also presented by Dupre (1993). More useful is the

concept that the essential purpose of science is to create mental maps to guide perception and action. As a result, there is a mutual participation between nature and consciousness.

In addition, thought is a collective phenomenon. As with collective properties or particles (e.g., electrons for electricity), thought must be coherent across individuals to be productive. Senge presents that Bohm says (1965) dialogue needs three conditions: (p. 26):

1. Everyone suspends assumptions.
2. Everyone is regarded as an equal colleague.
3. A Facilitator maintains context of the discussion.

Resistance wastes energy, just as electric resistance wastes energy (heat). All views are based on assumptions that should be identified. Unfortunately, the flow of dialogue is blocked by accepted assumptions. Suspending assumptions unravels the reasoning behind abstractions. Viewing all ideas as colleagues makes a positive tone to allow vulnerability. Everyone must want benefits of dialogue more than privileges of rank. In dialogue, complex issues are explored. In contrast, in discussion, decisions are made, so both must be balanced in discourse. Common ground must be found on which to agree. Personal vision provides a foundation for shared vision, just as reflection and inquiry are foundation for dialogue and discussion. The essence of visioning is that shared vision arises from personal visions. Organizations can be viewed as Living Systems (Senge, 2006, p. 267). Organizations are communities, not just production machines. (p. 267). Dialogue implies interaction and listening, while discussion, on the other hand, has the sole goal to win a point. Knowledge is created in a social process (p. 270), involving what we know how to do, and things done with one another. People work together to create value. The goal is to understand how one's own work occurs, and how to explain it to others.

Rise of the Archetypes

To explain how patterns of behavior develop in business, Senge (2006) built on the ideas of dialog and team learning to describe a set of nine underlying business archetypes. The inner workings of archetypes are presented in system dynamics terms as causal loop diagrams with interlinking loops, both reinforcing and balancing types, that illustrate behavior interactions with innate delays. In many cases, the structure of a specific **archetype describes a recurring operation that is likely not desired, but in which the business operation is stuck, and cannot escape**. Essentially an archetype denotes a specific defined business problem. The goal is to recognize the archetype as a recurring problem, and add a new path to the structure that allows a more preferred solution.

The original Shifting The Burden or goal archetype implies that problems arise from the actions that only solve symptoms (Senge, 2006, p. 391). A solution only works in the short-term with some immediate positive results, but in the long-term is not sufficient. The ineffective attempted solution is repeated at the expense of an actual fundamental solution that would work in the long-term. Unfortunately, eventually the actual fundamental solution may no longer work either, as the symptomatic solution is blindly, repeatedly applied. Example: selling more to existing customers rather than increasing market share by selling to new customers.

Although it sounds like a fit, shifting of burden does not mean to make dynamic changes in a system model to find creative new solutions matched to the context observed, as would contrast with that originally presented by Senge (2006). Rather, the ‘Shifting The Burden’ archetype points to the fact that unproductive behavior tends to arise in the long-term from some behavior

that seemed to be working in the short-term, while it actually is not, but just treats symptoms, while not treating the core problem. Senge (2006) contends Tragedy of the Commons combines Limits to Growth and Shifting of Burdens/Goals.

Similarly, the Limits to Growth (LTG) archetype refers to negative factors that slow or halt desired growth of a system. In studying this (and all) archetypes, the lesson is to find the element that is creating the problem, and choose actions that reverse or correct the undesirable looping of the archetype. Senge (2006) explains this in more detail for the LTG archetype since it may negate momentum to vision (read: Anticipation). Vision spreads as a reinforcing loop (p. 211), however, limiting factors slow down the desired vision cycle (balancing). Such limiting factors can be:

1. Viewing different ideal futures generates unmanageable conflicts.

   Based on increasing diversity and polarization.

   Leverage (solution) lies in identifying and understanding a limiting factor.

   Finding a common vision solves this.

2. Discouragement (like polarization) may create the limit.

   Capacity to keep creative tension may be lost, and thus effort.

   Encourage personal mastery to foster sustained commitment to vision.

3. Overwhelmed workers also lose focus on the vision.

   To solve this, pursue new vision vs. focus on managing current reality.

4. Forgetting connections to the others/team.

   To solve this, encourage reflecting and sharing on personal and overall vision.

Vision (or what might be viewed as longer term anticipation) becomes a 'living force' when people truly believe that they create their own future (p. 215). Managers must see themselves as creating current reality, and that they contribute to changing that reality. Unfortunately, a major threat to the anticipated vision is the concept that we can't change reality. In contrast, Senge contends the recognition of constraining archetypes empowers teams to change reality for the better for desired task achievement.

Enhanced System Archetypes

Systemic Thinking

Systemic (Systems) Thinking behavior is created from loops having interrelated variables (Flood, 1999, p. 85). Complexity Theory is part of Systemic Thinking. (p. 85). Flood contends that complex interrelationships and emergent behavior are unknowable (p. 87). This builds on the ideas from Senge (2006) that explain thinking is a process that leads to thoughts (or assumptions). From this, reflection and action are linked, leading to a goal. Thus, proper use of team reflection takes action towards a desired goal or task achievement.

Structure includes organizational functions and forms of coordination, communication, and control. (p. 104). Bureaucracy is the establishment of lines of authority, both rational and legal (p. 105). The whole organization, like a whole organism, contains structure and lines of communication to impart authority for action to all sections of the organization.

Whole Organism

A whole organism is more than the sum of its parts. (synergy: p. 29). Existence of an organism cannot be properly understood merely as the behavior of some fundamental parts. An organism co-exists in relation to its environment (niche). Function (operation) and structure diversity are maintained by continuous flow of energy and information between organism and environment. (p. 106). Open Systems Theory (OST) (also called organismic biology) uses functional (operational) and relational criteria rather than a reductionist approach that analyzes fundamental parts (organism with environment). An organism is a complex thing of many interrelating parts, resulting in a whole with overall integrity. It includes several processes:

Self-organization: attained through differentiation.

Equifinality: reaching a final state from initial conditions.

Teleology: behavior for a purpose 'known in advance' (implied anticipation).

OST contrasts with closed systems thinking (ClST) that contends organism behavior is based on fundamental principles and laws just as physics. Emphasized is efficiency and effectiveness of interacting parts.

For ClST, the organism is a mere machine of simple interworking parts. Success (in ClST) arises from repetitive performance of routine tasks to produce a single specific product. (Senge ignored this). In addition, drudgery and demotivation arise from mindless activity. OST, on the other hand, contends physics is unable to 'appreciate' dynamics of organization, so parts are better studied as a whole. The organism is open to the environment, with survival via transforming inputs and by adapting to change. Parts of a society (people) have needs; motivation enriches satisfaction and productivity, including democracy and autonomy.

OST actually is the forerunner to General System Theory (GST) (p. 31). GST envisions a unity of science, formulating and deriving principles that apply to all systems in general. The Society for General Systems Research (SGSR) exists today; (initially it was: Society for Advancement of General Systems Theory.) The founding aims were:

1. Investigate isomorphy (similarity) of concepts/models across fields.
2. Develop theoretical models for various fields.
3. Minimize duplicate model efforts across fields.
4. Improve communication among specialists for unity of science.

Senge (2006) suggested systemic thinking used archetypes to these aims. Von Betanlanffy saw specialization as breaking down integrated science, and sought to prevent closed/isolated research in specialized fields. Rosen (1979) (who developed the Rosen Model for congruent systems) compared similarities across systems, showing that one can use system A to learn about similar system B.

Systemic Thinking

Interrelatedness infers everything is interrelated with everything else (Flood, 1999, p. 91). Thus, systemic appreciation is an ever-expanding activity. Complexity Theory (CT) adds the insight that emergence results from dynamics and spontaneous self-organization. In contrast, Chaos Theory tries to explain that even a butterfly can change the weather worldwide. Thus, the total dynamic is even more complex by CT. Any comprehensive systemic representation is overwhelming in interrelationships and recurring emergence. (p. 92). A further distance from a locality in space and time blurs interpretation of events. Thus, people only can deal with items close to them in space and time. This leads to bounding formulation/appreciation in local terms.

All interpretations are only partial and temporary views. Flood states that Churchman contends bounded appreciation is most relevant and acceptable. In addition, boundary judgements include a debate on ethical issues and dilemmas. Each boundary is a choice for who benefits, and who does not. Thus, setting boundaries has a moral implication. A key principle of systemic thinking is to remain ethically alert to choices. Boundary choice also sets a purpose to pursue, and values. For example, unemployed people are excluded from organizational analysis. Thus, perpetual unemployment is not recognized. Including the unemployed in such an analysis opens up equal opportunity issues. Boundary judgements create an action area that is partial and temporary, where systemic appreciation may deepen for a specific idea or item. To summarize this view, a boundary must be set for any system, and setting that boundary is a moral decision as to the purpose of the system and possible behaviors that may be included for successful observable task achievement.

Limits To Growth Archetype Structure

A formal process program code was developed by Hayward and Boswell (2014) to represent the Limits to Growth archetype. It included the interaction of dynamic looping to capture the overall priority execution of the items in related and overlapping loops. The work was built on a simple set of four atomic behaviors that represented the responses of reinforcing and balancing loops (Fig. 9).

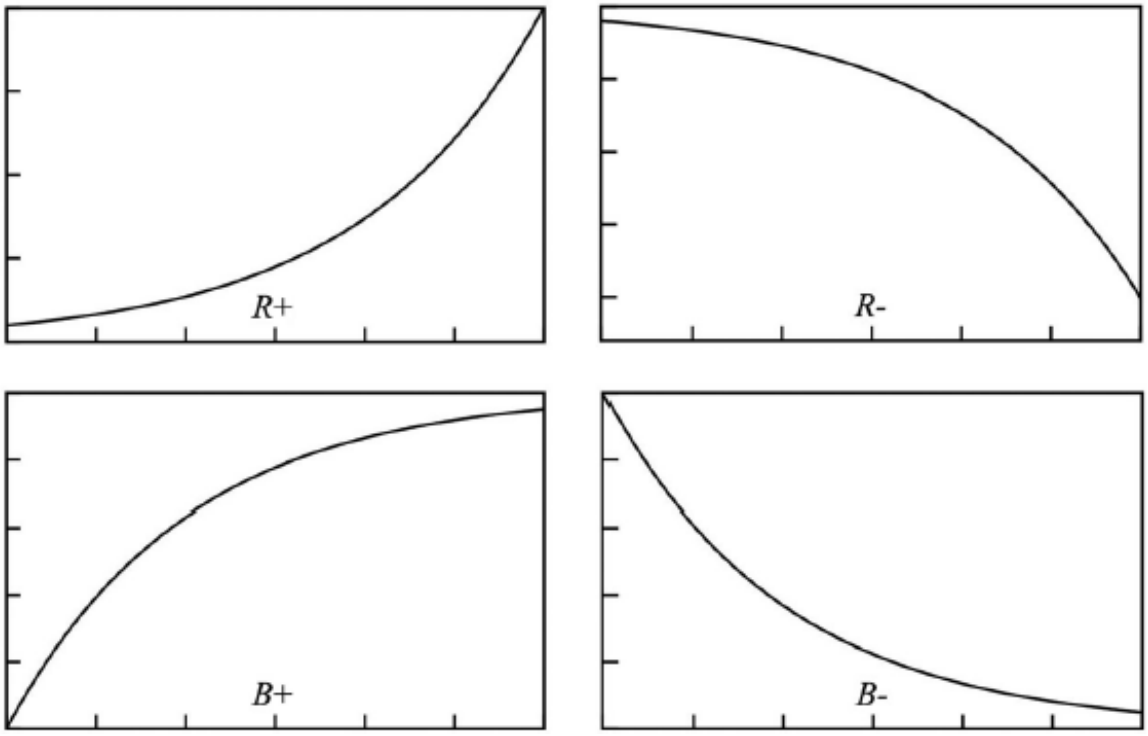


Figure 9. Atomic behaviors of convergence observed in system dynamics loops as reinforcing (R, unstable) or balancing (B, stable) having trends as increasing (+) or decreasing (-). Individual or combinations of loops will exhibit these nonlinear trends (from: Hayward and Boswell, 2014, Fig. 1).

The underlying differential equation for the Limits to Growth archetype is captured in a stock and flow map that relates various elements in the system to key limiting factors (Fig. 10.). The specific differential equations for the process for the change in *x* (derivative with time):

$$dx/dt = \dot{x} = ax\,(1 - x/M) - bx \qquad (2.4)$$

and

$$x(t) = x(t\text{-}1) + \dot{x} \qquad \text{(Euler integration)} \qquad (2.5)$$

Recall the definition for equations within the stock and flow map is (Sterman, 2000, p. 194):

d(Stock)/dt= ΔStock= Inflow(t)-Outflow(t)

$$\text{and thus: Stock}(t) = \text{Stock}(t\text{-}1) + \int_{t-1}^{t} [\text{Inflow}(s) - \text{Outflow}(s)]\, ds \qquad (2.6)$$

Thus, formulation of the equation in process program code is representative of the relations show in the stock and flow map.

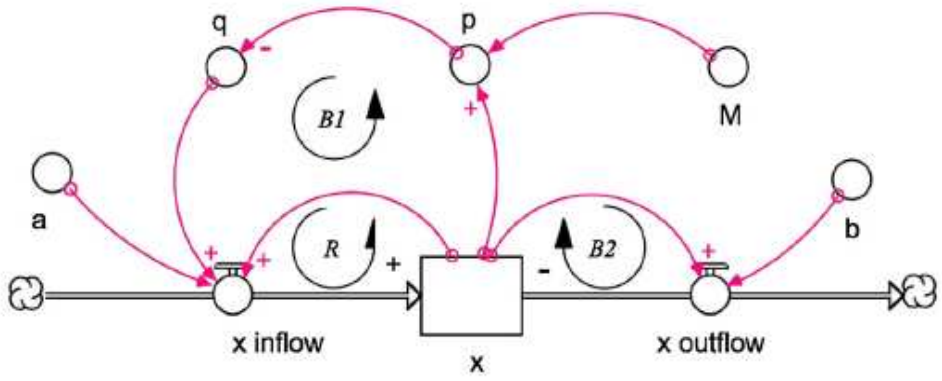


Figure 10. Simulation stocks & flows map for Archetype Limits to Growth as a first order model (one variable only: x) (from: Hayward and Boswell, 2014, Fig. 2, related eq. 6).

Difficulty arises in that the arrows do not represent a specific operation (could be a sum or product). In the map (Fig. 10), M => p means p= x/M, while p => q means q= 1-p, so adding a unique symbol is clearer. These conventions change between sources to make it more confusing. Also, in a causal loop diagram, the sign (+/-) merely indicates the polarity (positive direct or negative opposing), so the symbols and arrows are even more confusing. Hence, Hayward and Boswell developed the process program code to remove ambiguity from such unclear representations, and also presented resulting graphs of output for the various processes.

Sterman (2000) points out that other work by Richardson points out pitfalls of causal diagrams, most serious that of not distinguishing between stocks or flows. Even more importantly, Dowling, MacDonald, & Richardson (1995) label causal loop diagrams as ‘inherently weak’, not distinguishing conserved vs. information flows, and unclear as to hidden flows, net rates, and parameter limits. Thus, causal loop diagrams are limited for presenting structure and behavior

clearly. Instead, actual specific computer language code is needed, and multiple versions may seem to represent the same causal loop diagram. Thus, one must refine such diagrams to code that is clear as to intent and functional results.

Dowling et al. (1995, p. 455) contend causal loop diagrams are too general, so specific computer code must be created to embody the desired relations and behaviors:

> "While causal loop diagrams are a simple tool to use in communicating system structure, they are inherently weak because they do not distinguish between conserved flows and information flows. As such, they obfuscate [confuse] direct causal relationships between rates and levels. This is an important factor since most of the structures presented by Senge and by Wolstenholme and Corben represent conserved flows."

Also they present the following (p. 455):

> "Further, it has been argued by Richardson… it is impossible to determine behavior simply from loop polarity because loop polarity does not create behavior. Rather it is the rate-level structure that determines behavior. The fact that causal loop diagrams do not reflect important factors such as hidden loops, net rates, and parameters further limits their ability to provide a clear understanding of structure and behavior."

Shifting The Burden (Goals) Archetype Structure

The original Shifting The Burden or goal archetype implies that problems arise from the actions that only solve symptoms (Senge, 2006, p. 391) (Fig. 11). A solution only works in the short-term with some immediate positive results, but in the long-term is not sufficient. The ineffective attempted solution is repeated at the expense of an actual fundamental solution that would work

in the long-term. Unfortunately, eventually the actual fundamental solution may no longer work either, as the symptomatic solution is blindly and repeatedly applied. Example: selling more to existing customers rather than increasing market share by selling to new customers.

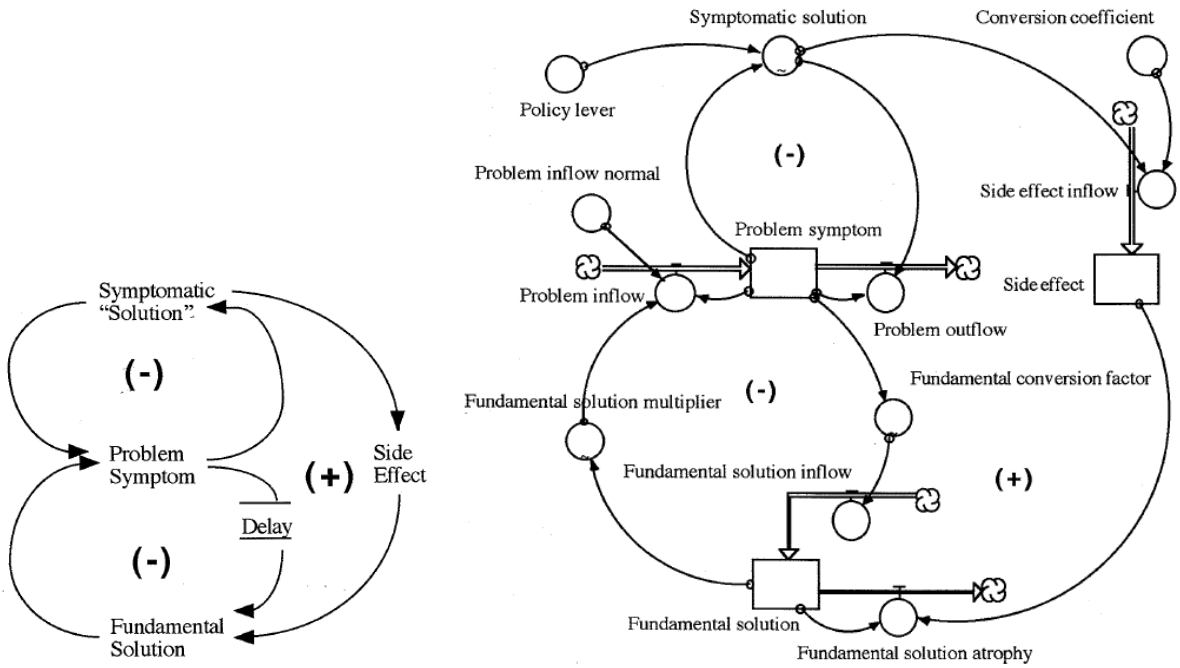


Figure 11. Shifting the Burden archetype causal loop diagram (left) and stock and flow map (right) evidencing the degree of detail added to clarify beyond the causal loop diagram. The system contains two stocks as a second order system, and one pipeline delay (outflow=inflow after delay time) shown as a separate stock. Two negative feedback balancing loops and one dominant positive reinforcing loop interact to create the system dynamics (From: Dowling et al., 1995, Figs. 2A & B, p. 458 & 459).

Shifting of Burdens/Goals does not just make dynamic changes in a system to find creative new solutions matched to the context observed. Instead, the 'Shifting The Burden/Goal' archetype represents that unproductive behavior arises long-term from a behavior that seemed to be working in the short-term, while it actually is not. It only treats symptoms, not the core problem.

As with the Limits To Growth archetype, specifics for the Shifting The Burden archetype must be specifically given in detailed processing code to understand the actual operation of the underlying loops and delays to create the behaviors (Fig. 12).

Equation List For "Shifting The Burden"

```
Fundamental_solution(t) = Fundamental_solution(t - dt) + (Fundamental_solution_inflow -
Fundamental_solution_atrophy) * dt
INIT Fundamental_solution = 0
INFLOWS:
Fundamental_solution_inflow = Fundamental_conversion_factor
OUTFLOWS:
Fundamental_solution_atrophy = Fundamental_solution*Side_effect
Problem_symptom(t) = Problem_symptom(t - dt) + (Problem_inflow - Problem_outflow) * dt
INIT Problem_symptom = 100
INFLOWS:
Problem_inflow = Problem_symptom*(Problem_inflow_normal-Fundamental_solution_multiplier)
OUTFLOWS:
Problem_outflow = (Problem_symptom*Symptomatic_solution)+(Problem_symptom*.1)
Side_effect(t) = Side_effect(t - dt) + (Side_effect_inflow) * dt
INIT Side_effect = 0
INFLOWS:
Side_effect_inflow = Conversion_coefficient*Symptomatic_solution
Conversion_coefficient = .05
Policy lever= 1
Problem_inflow_normal = .2
Fundamental_conversion_factor = GRAPH(SMTH1(Problem_symptom,1))
(0.00, 0.00), (20.0, 0.01), (40.0, 0.02), (60.0, 0.03), (80.0, 0.04), (100, 0.05), (120, 0.062), (140, 0.077), (160,
0.098), (180, 0.117), (200, 0.133)
Fundamental_solution_multiplier = GRAPH(Fundamental_solution)
(0.00, 0.00), (0.1, 0.00), (0.2, 0.001), (0.3, 0.0163), (0.4, 0.035), (0.5, 0.0663), (0.6, 0.0963), (0.7, 0.126), (0.8,
0.152), (0.9, 0.182), (1, 0.2)
Symptomatic_solution = GRAPH(Problem_symptom*Policy)
(0.00, 0.00), (20.0, 0.0675), (40.0, 0.155), (60.0, 0.188), (80.0, 0.228), (100, 0.268), (120, 0.295), (140, 0.34),
(160, 0.405), (180, 0.46), (200, 0.492)
```

Figure 12. Shifting the Burden archetype process code with equations and tables to represent system dynamics (from: Dowling et al., 1995, Table 3).
One equation is not on the previous flow map, so is an anomaly:
ProblemOutflow= (ProblemSymptom * SymptomaticSolution) +
(ProblemSymptom * 0.1)

For this specific construct of the Shifting the Burden (STB) archetype, typical system dynamics programing defines the inflows, outflows, and values for each of the stocks in the system. The important consideration here is that the underlying relations suggested in the causal loop map

and the stock and flow diagram are represented as conceptual equation relations that can more specifically express the connections and relations that will reflect the actual operation of the archetype, and thus captures more completely the behaviors to be observed as a result of operation of the archetype. As an added item, the inclusion of table lookup functions (shown as GRAPHs at the bottom of the code) provide for specific direct relations between essential components within the overall looping structure. Therefore, the less obvious relations are represented in designated calculations that should lead to reproducibility of the results by creation of another instance of the STB archetype representation. Representation of the STB archetype code is much preferred to the mere presentation as loop diagrams with poorly defined interations between the components.

Archetype Game Irony

Complex systems are difficult to understand, and the underlying generic structures might not be recognizable, as was demonstrated by a system dynamics business game simulation developed and tested with participants by Bagodi and Mahanty (2015). The game was based on a Shifting the Burden archetype structure for an architecture (Fig. 13 & 14), where symptomatic solutions tend to be chosen by the participants (acting managers) rather than the preferred fundamental solution, mainly because the system is generally too complex to recognize both the underlying system archetype and the need for the alternate preferred solution. Unfortunately, the symptomatic solution allows for perceived minor improvements in the short run (hire more salespeople to increase sales) but fails to work over the long run where a more fundamental solution is needed (improved product quality via attractiveness) (Fig. 13). [This is ironic, since the choice for a symptomatic solution (creating failure) contradicts what is expected (success). It

is *not* a paradox, since a paradox is a statement that contradicts itself, such as: “I always lie.” See: http://english.stackexchange.com/questions/344071/the-difference-between-irony-and-paradox] None of the participants that played the game was able to recognize either the archetype or the preferred fundamental solution, and hence did not attain the desired outcome of a successful growing business.

Indeed, the results of this study quite plainly illustrate how one can operate within a system with underlying operation of that of a know archetype, yet not recognize the principles at work. In that sense, we all may be like fish living in the water, and not realize the importance and limitaions of the water to fundamental operations for our existence. The case also suggests one must look closely and critically at overall operations in attempt to tease out and recognize the mechanism at work, and make concerted effort to identify what the underlying archetype might be. Once identified, the know aspects of the defined archetype should assist in leading to understanding of system operation, pitfalls, possibilities, and means to alter the operation that may lead to desired or more meaningful results. The is no simple avenue, as this case shows. Instead, one must become familiar enoughwith individual archetypes to map the components of a system onto the conceptual description of the archetype. Only then can the archetype become a tool that can be used for further understanding and analyses.

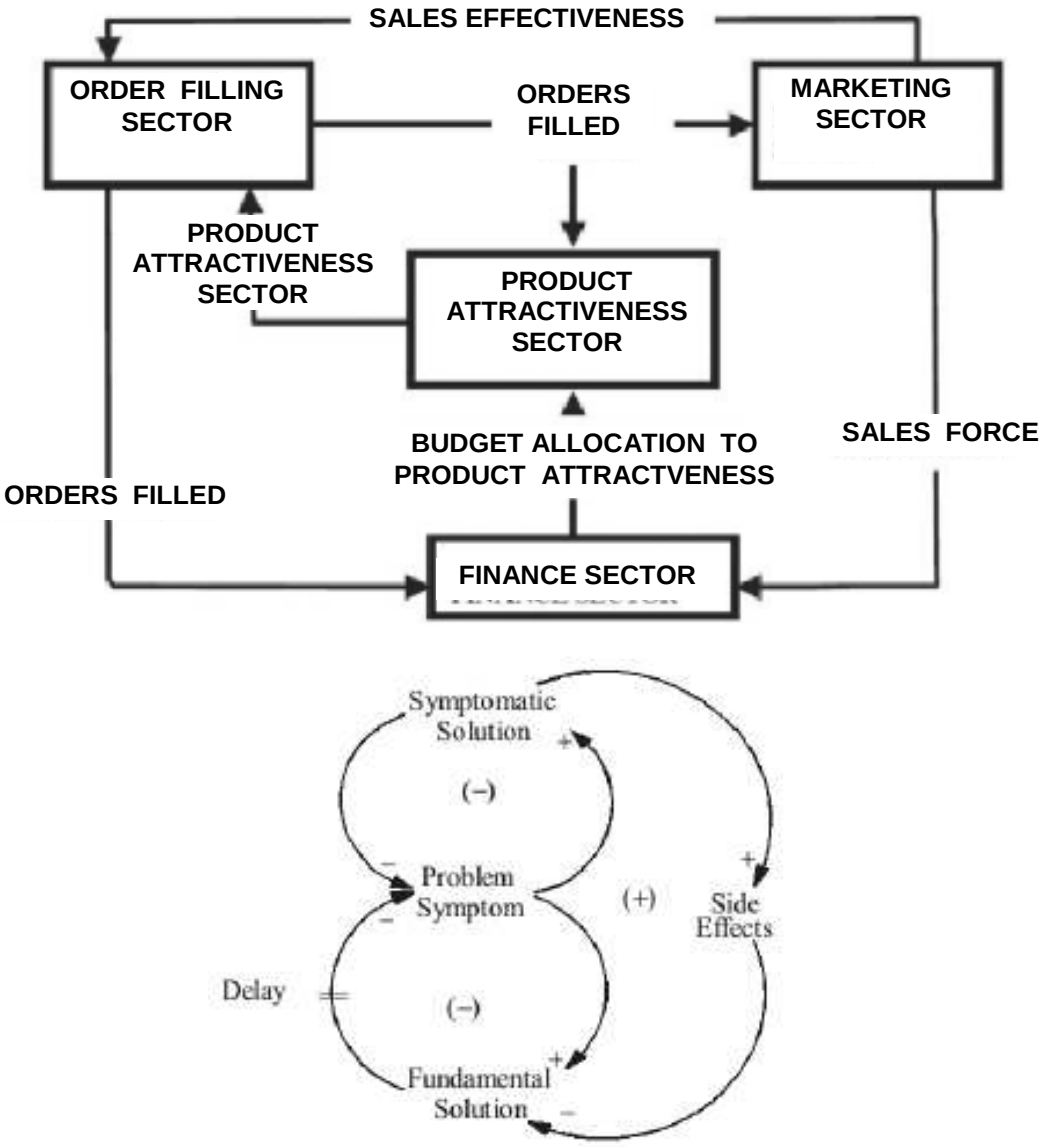


Figure 13. Sectoral overview diagram (top) and underlying Shifting the Burden archetype (bottom). Note there are two balancing loops (negative signs) and one reinforcing loop (positive sign) of the side effects (Bagodi and Mahanty, 2015, pp. 385 & 387, Figs. 1 & 2).

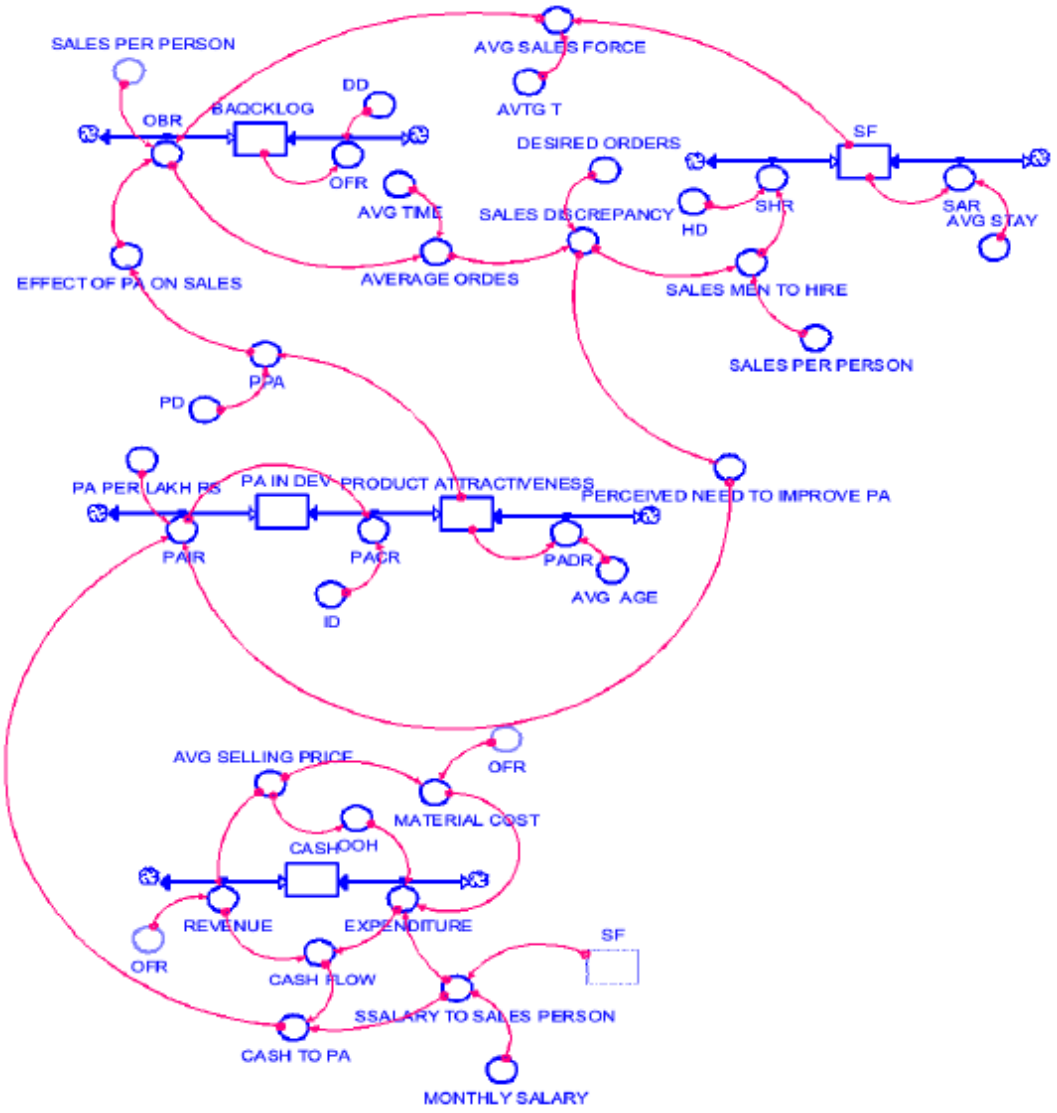


Figure 14. Flow diagram of the simulation model for the Shifting the Burden archetype business system game. The unpreferred short-term solution hires more salespeople, and eventually fails, while the preferred long-term solution is to increase product attractiveness, which in turn increases sales, and is thereby successful (Bagodi and Mahanty, 2015, p. 389, Fig. 3).

Simplified Basic Archetypes

There are around 10 recognized system archetypes, depending on the source consulted (Senge, 2006; Wolstenholme, 2003, 2004). Each archetype generally defines an undesirable operation for a system (problem) that can be corrected if a proper solution link is introduced, and generally must pass across perceived system boundaries (Fig. 15) (Wolstenholme, 2003, Fig. 1). Both intended and unintended consequences are involved. Four distinct two loop system archetypes have been identified, using the four possible combinations of reinforcing and balancing loops, that can represent all the other more specialized archetypes: 1) Underachievement, 2) Out of Control, 3) Relative Achievement, and 4) Relative Control (Wolstenholme, 2003, p. 11) (Fig. 16).

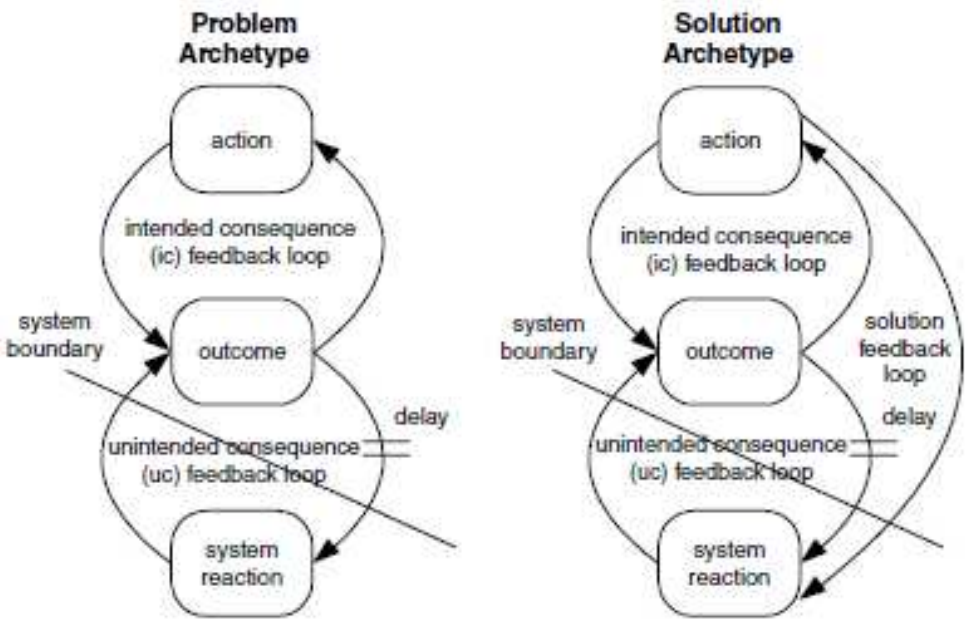


Figure 15. Flow diagram structure of the Problem (left) and Solution (right) totally generic archetypes, indicating the presence of intended and unintended consequences (from: Wolstenholme, 2003, Fig. 1).

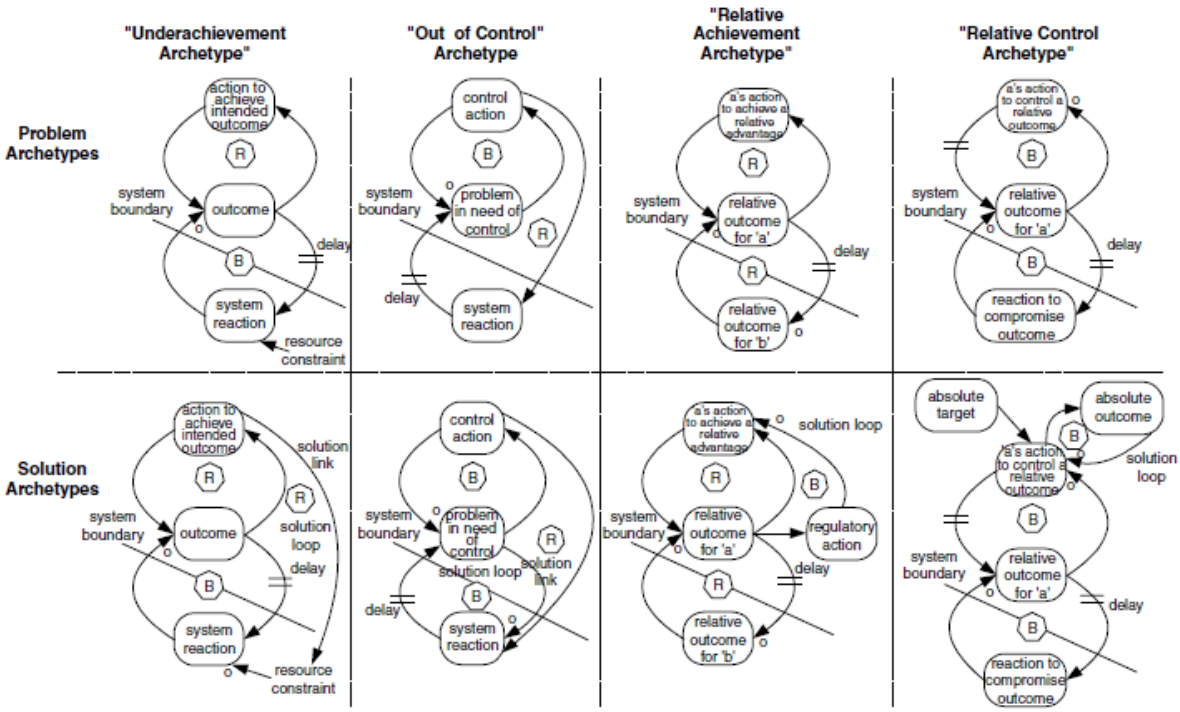


Figure 16. Set of four flow diagram structures for both the Problem and Solution totally generic archetypes, indicating the presence of intended and unintended consequences. R= reinforcing loop. B= balancing loop. O= opposing action (from: Wolstenholme, 2003, Fig. 2).

Each totally generic archetype represents one or more specific archetypes based on the relations of the loops, with both intended and unintended consequences (Wolstenholme, 2003, Figs. 3 to 6).

1. Underachievement generic archetype.

    Archetypes (three different ones): Limits to Growth/Success, Tragedy of the Commons, and Growth and Underinvestment.

    Loops: reinforcing as intended consequences, balancing as unintended consequences.

2. Out of Control generic archetype.

    Archetypes (three different ones): Fixes That Fail, Shifting the Burden, and Accidental Adversaries.

Loops: reinforcing as unintended consequences, balancing as intended consequences.

3. Relative Achievement generic archetype.

Archetypes (one only): Success to the Successful.

Loops: two reinforcing as either intended consequences and unintended consequences.

4. Relative Control generic archetype.

Archetypes (two different ones): Escalation, and Drifting Goals.

Loops: two balancing as either intended consequences and unintended consequences.

Systems thinking using system dynamics must recognize system boundaries and make them as transparent as possible. Adding a path to the system that crosses the boundary is a crucial step to provide a solution to the archetype identified (Wolstenholme, 2003, p. 25 & 26; Wolstenholme, 2004).

ROBOTIC ARCHITECTURES

Subsumption Architecture

The earliest work in BBR used a computation model called subsumption architecture (SA) that aimed to program intelligent, situated, and embodied robotic agents based on well-defined robotic principles which follow (Brooks, 1999, p. 172). Computation is organized as asynchronous networks of active elements with fixed topology and unidirectional connections. Messages through connections have no implicit semantics, and meanings are dependent on dynamics built into both sender and receiver. Sensors and actuators are connected asynchronously to the network. The goal is to study complete integrated intelligent autonomous agents embodied as mobile robots, and situated in the research lab world and operate in real

time; environments should not be changed in any way for the robots. No central model of the world is maintained; all data are distributed over computational elements. There is no central control, and there are no separate systems for perception or actuation. There is no hierarchy; results of needed computations are available on input lines with no synchronization of production and use of messages. The user piece of the network uses whatever information is available at the time it is required. All layers (or behaviors) run in parallel, with each layer as a behavior, and a conflict resolution method may be needed to resolve differing commands. The world is used as a communication medium for processes within a single robot.

Connell (1990, p. 154) used a colony architecture that was expanded from the subsumption architecture of Brooks (1999) to build an autonomous robot, Herbert, that collected soda cans. Colony architecture had 1) independent agents: modularity for each agent to be independent so any agent can be replaced or upgraded independently, and 2) local control: information collection and decisions for control should be in temporally local modules (p. 12). There is no internal world model, and the world is used directly as a model of itself for direct sensing and acting. Simple systems are combined with a behavior fusion to obtain the same result as fusing multiple sensors (p. 14). The local control dictates a means of reactive programming that responds to specific events in the world. Connell (1990) compared colony architecture with four other architectures, emphasizing advantages of the independence and local control features of Herbert that were effective through suppression of lower level behaviors by higher ones. For the more current application of anticipation to robust robot design, the advantages of colony architecture are apparent to use independence and local control to attain desired hierarchy by

behavior suppression, and could be included to provide for effective application of anticipation to overall agent control.

Robotic Principles

A set of eight Agent Design Principles have been defined that are similar to those of Brooks (Pfeifer & Bongard, 2007). The overall three-constituents principle stipulates that designing an intelligent agent involves the interaction between an ecological niche (environment), desired tasks and behaviors, and a specific robotic agent. A complete agent must behave in the real world and is built to exploit properties of a specific ecological niche so that interaction with the niche will have a much easier (cheaper) design and construction. For redundancy, intelligent agents have different subsystems function based on different physical processes, and a partial functionality overlap exists between different subsystems. Sensory stimulation is induced through sensory-motor coordination. For ecological balance in a certain task niche, a match exits between complexities of the sensory, motor, and neural systems, with a balance between morphology, materials, and environment. Intelligence emerges from a large number of parallel processes that are often coordinated through embodiment, and embodied interaction with the environment. A value system is included that contains a basic set of assumptions about what is good for the agent. These agent design principles, along with those from colony and subsumption architectures, provide a strong basis for designing robots that could use the principles of anticipation to operate more robustly in the real world. Indeed, the focus here is to use a principled approach that matches behavior directly to perceived niche conditions to result in desired tack achievement.

BEHAVIOR

Task, Niche Environment, and Agent

Behavior is the observed response of an organism or robotic agent to current niche environmental conditions to create task achievement. This definition of behavior was extended from three previous ones of Nehmzow (2000), Brooks (1999), and Malcolm and Smithers (1990). A robot agent can be a specific physical robot or programed process that performs a behavior to attain desired task achievement.

Previously for behavior-based robotics (BBR), Nehmzow (2000, p. 10) contends behavior of a robot is dependent on the task to be achieved and the environment the robot is in, and the three elements influence each other (see beginning of Chapter 2, Fig. 1, p. 7). It is represented as the task-environment-agent (TEA) triangle that interlinks the three dependent elements. Nehmzow credits Smithers with the example of a spider that survives in the outdoors, but is 'incompetent' in a bathtub. This spider example shows how an organism (animal or robot agent) is best fit to a specific task in a specific context (niche environment), and that organism or agent will not necessarily operate correctly in a different context. The organism or agent is matched to its niche environment, with behavior coupled to the niche conditions. Hence, Nehmzow thinks a general purpose robot cannot exist, but each robot has the specific purpose or a task to be achieved. Thus, functional operation defines the behavior of a robot in an environment for the specific task. Defining all three TEA elements together simultaneously defines the agent completely.

Another similar version of the TEA behavior explanation is presented by Brooks (1999). An organization of subsumption architecture (SA) uses successive incremental layers of

programming linking sensing to action. Each layer operates autonomously by assuming control from lower levels, connecting the real world with lower levels. The result is a simple type of distributed control. The SA is based on a collection of individual processors, each an augmented finite state machine (AFSM), with simple messages passing between them, permitting activation or deactivation of each AFSM as dictated by the sensed environment (Brooks, 1999, p. 40). Multiple AFSM processors are grouped to perform a behavior (Brooks, 1999, p. 41). Each behavior acts as an internal abstraction barrier, so that each behavior (group of AFSMs) is separate from another (a different group of AFSMs). A *behavior* is thus a group of AFSM processors between which simple messages are passed, causing suppression or inhibition between processors both within the behavior group, or passed messages to suppress or inhibit other lower level behavior as other processor groups.

A robotic assembly system, SOMASS, by Malcolm and Smithers (1990) included behavioral modules that encapsulate useful elements of specific behaviors to be

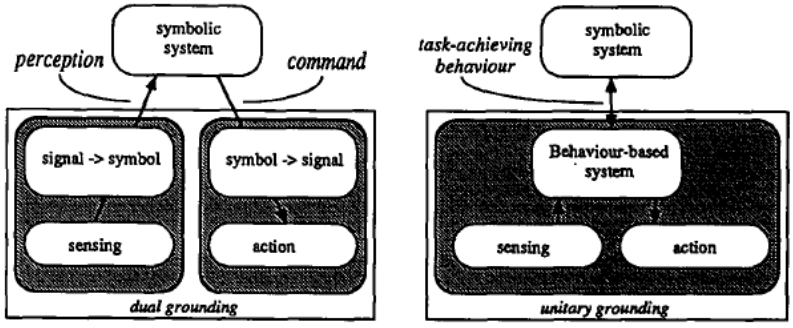


Figure 17. Two kinds of grounding: Dual (left) verses Unitary Grounding (Malcolm & Smithers,1990).

executed by the assembly system. These behavior modules are used by an overall cognitive system to achieve desired behavior for functional capability. The behavioral modules ground the behavior in the real world through a behavior-based system using unitary grounding that places

both abstraction and effective capability in a single behavioral module, in contrast to dual grounded system that grounds abstraction and effective capability in two separate units instead of one, as Malcolm and Smithers contend occurs in the Explicit-World-Model (Fig. 17). The behavioral modules are considered the subcognitive part of the system. Thus, the hybrid system constructed for SOMASS uses an overall cognitive system to direct the use of subcognitive behavioral modules to perform desired behavior. The term hybrid implies two kinds of components are in an autonomous system: cognitive (traditional Knowledge-Based from classical artificial intelligence, AI) and subcognitive ones (Behavior-Based as described by Brooks).

Social Robotics With Theory of the Mind

Social robotics has earned increased interest in recent years. Breazeal (2000) and Scassaletti (2001) explored ways to add social behavior to robots. Breazeal (2000) developed a social robot named Kismet that recognizes facial gestures and simple words (auditory signals) from humans to cue behavior that created facial expressions on Kismet, thus encouraging people to interact with Kismet like it is an infant. Going further, Scassaletti (2001) based his approach on the Theory of the Mind from philosophy to attempt to incorporate behavior and thinking into the robot processes to resemble the thinking of humans. Though they both used the small robot Kismet for their work, similar to a human infant in size, the background technology and electronics enlisted the support of a bank of over 10 computers in an adjoining room to allow the reactions of Kismet to humans to proceed. Thus, in structure as well as function the small robot agent was considerably larger than life, and one might say the observed behavior responses were not entirely just in the observed robot Kismet. Further work by Breazeal has developed a more

compact Jibo robot (a social family robot) that is stationary for home use, and strives to provide everyday informational support for typical technology laden human lifestyle (https://www.jibo.com). All these cases have moved beyond the basics of a subsumption architecture to more elaborate interacting descriptions that could be said no longer are strictly behavior-based robotics.

ANTICIPATION FOR BEHAVIOR-BASED ROBOTICS

An anticipation set contains a small number of specific well-defined behaviors that can be manifest as a choice for the agent that matches the current niche condition. A niche is a specific environment where the behavior of an agent is expected to have successful task achievement. The niche may afford for more than one behavior, yet one niche condition is best matched to one specific behavior at a current time. Fitness is a measure of how well a behavior is matched to the niche environment. The best value of fitness cues a specific choice of behavior. Percepts are perceptions of the niche environment by the agent to determine what behavior is best afforded to the current niche condition. Threshold coupling uses the value of fitness to cue selection of a choice behavior from the overall anticipation set.

Percepts arise from the area of psychology, where a percept is the mental recreation of a distal (external) stimulus. For robotics, the mental reference is replaced by an agent. A real world object is the distal stimulus or distal object. Through a physical process (light, sound, etc.) a sensory device/organ is stimulated, in turn using energy to create neural activity (called transduction). The internal raw pattern is the proximal stimulus, which is transmitted to the brain (agent processor) for processing. The resultant recreation of the distal stimulus in the brain is a

percept. Overall, perception is creation of mental representations (images or archetypes) using the proximal stimuli derived from distal stimuli. For example, a cat as a distal stimulus is detected by light energy entering the eye to form an image on the retina as a proximal stimulus, and the reconstruction of the image in the brain (agent processor) is the percept. A bird singing as a distal stimulus uses sound energy to move auditory receptors as the proximal stimulus, and interpretation by the brain (agent processor) is the percept. Intelligent agents choose to act both on individual and sequences of multiple percepts. An agent function maps each percept to an action, and subsequently to the next action. (From: https://en.wikipedia.org/wiki/Perception ; and https://en.wikipedia.org/wiki/Percept_%28artificial_intelligence%29).

For the purposes of this study, a percept is an abstract representation of an element or factor in the niche. Synonyms include: form, rule, habit, image, code, and covenance. In brief, a percept in the FS is the abstraction of an elemental factor in the NS. Anticipation acts by using percepts of the niche to cue a behavior that is manifest to produce a matching behavior by the agent, and lead to successful observed task achievement.

CHAPTER 3. ANTICIPATION IN ROBOTICS

OVERVIEW

The notion of anticipation involves understanding what anticipation is, what it looks like, how it operates, and how it might be included in artificial robot agent systems. The discussion begins with a historical perspective and description of how anticipation appears to work so that the future actually influences the present (or the past leading to the present). The key is for a living biological system, or an artificial robotic system, that can conceive of an outcome prior to its occurrence (Rosen, 1991; Rosen, 2012), to actually use the expectations about that outcome to manifest a choice of behavior that leads to a goal for desired task achievement. For the previous design model robot that operated as a wall follow (or MURAMATOR), and for the new design model TOURIST, the ultimate goal for task achievement is to both follow a wall and to avoid stationary stray objects. Using the congruence framework (Rosen 1991; Rosen 2012), an anticipation simulation model (ANSIM) is developed to study responses to various situations. The implications for properly adding anticipation to a working base simulation formal system model abstraction are described. The outcome of these formal system simulations will be described in more detail in Chapter 4. We begin the process by first looking at and defining in detail the traits of anticipation both in biological living and artificial robotic systems.

DESCRIPTION OF ANTICIPATION

Overall architecture of a robotic system using anticipation has been formalized using system dynamics for a mechanism of anticipation to cue direct behavior (Figure 18).

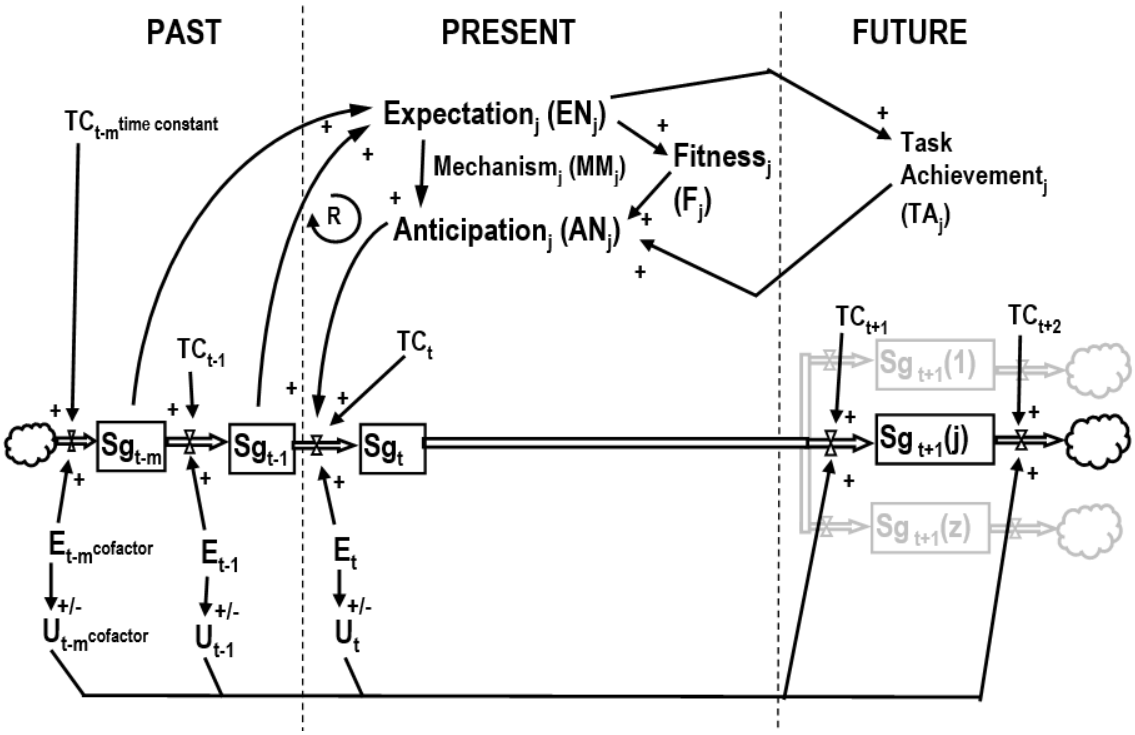


Figure 18. System dynamics model of how past stages of events create expectations of task achievement in the future, producing anticipation with associated fitness that permits choice to change present behavior and future events. Gray paths are not chosen.

Stages (Sg_i) in the past create expectations (EN_j) for many possible outcomes in the future. Each expectation for a specific future task achievement has some mechanism (MM_j) to create a specific anticipation (AN_j) with an associated fitness (F_j) that is calculated from invariant sensed percepts (p_1, …, p_n) in the environment (Figure 19 & 20), each with some weighting (a_1, …, a_n) based on past experiences. Only the minimum set of invariants that are needed in the contextual environment are included to afford a specific behavior with greatest fitness. The anticipation with maximum fitness is chosen to change present behavior and the one set of future stages that follow from it (the gray stages are not chosen, Figure 18). Sensed qualities in the environment are measured as cofactor E, and the level of related causal cofactor U is predicted that eventually directly influences future events (Rosen, 2012).

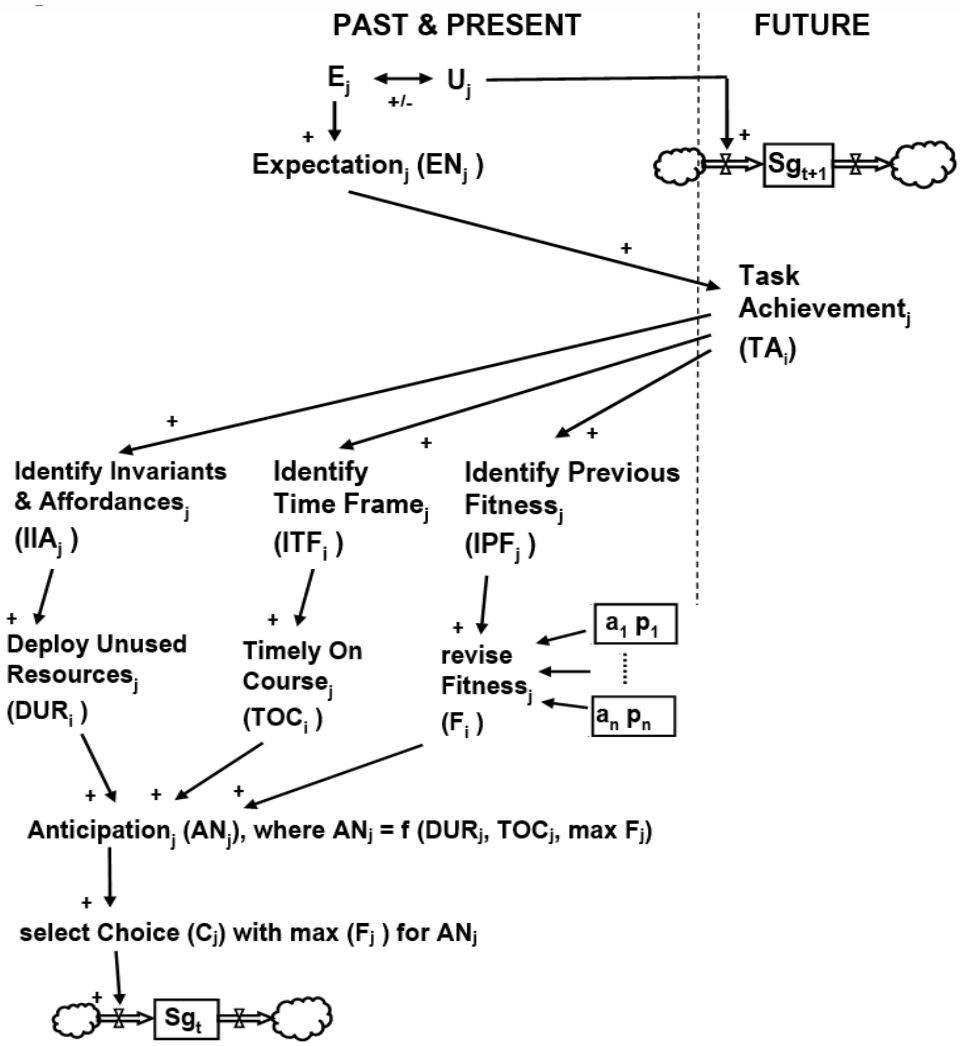


Figure 19. Dynamics of cofactors, E and U, for future task achievement using niche invariants to afford deployment of resources by anticipation and associated fitness to make a choice of present desired behavior (after: Sterman, 2000).

Task achievement is set in the future, and may differ for each associated expectation (EN_j), and causes specific elements to be identified that are required to reach that task, including invariants with their affordances, a time frame for action, and previous estimated value for a fitness. From these, resources are identified that need to be deployed along some timeline for action, and the fitness is updated based on any current percepts with some weighting as to importance for an organism relating to the task, and these differ from the environmental measurement, E, that

initially created the expectation. These items all combine to form some anticipation for a specific outcome, and the one anticipation with maximum fitness prompts selection of a choice that cues a behavior in the present time stage.

A revised fitness value is calculated from current relevant percept inputs, as invariants in the niche, using weightings of importance to the organism and task achievement that promote or inhibit fitness (Figure 20). After some time constant delay coupled with dynamics of the process, the calculated fitness, F_j, is compared to some threshold value based on previous experience or knowledge, and a high enough value or appropriate range can cue a discrete action (on/off) output (one that is discontinuous or nonlinear), or a continuously variable action choice (magnitude of C_j) inheriting its value from the derived fitness itself. The cued choice selects a specific desired behavior that is afforded by invariants as matched to the contextual environment, and ensures deployment of needed resources in a timely scheduled manner with the current maximum desired fitness (Fig. 20). This overall architecture based on anticipation is considered able to extend the colony architecture that revised basic subsumption architecture from the founding of BBRs (Connell, 1990; Brooks, 1999).

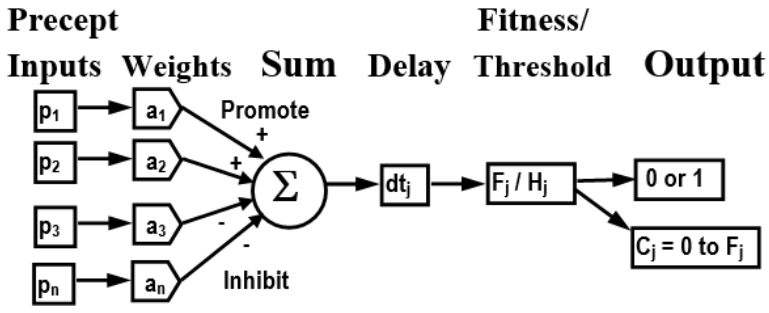


Figure 20. Weighted percepts join with delays to calculate fitness to compare with thresholds that create discrete or continuous outputs. (modified from: Connell, 1990).

RECOGNIZING ANTICIPATION

For humans, and some animals and plants in biology, it is possible to view behavior that is readily labeled to exhibit anticipation (AN). For example, scheduling of events by humans from business meetings to sports events, are all longer term displays of anticipation. For animals and plants, building a nest to rear young or forming flower buds during the summer that will bloom the next spring are similar scheduling examples (Rosen, 2012; Nadin, 2002). Shorter term displays of the notion of AN are desired for robotics, since the time frame for action and speed of operation are generally in a briefer time that the biological examples above. This leads to the questions of what does AN involve, how is AN recognized or observed, and how can it be included in robotics.

A discussed above, AN relates to an expectation about the future, either by some neural system in biology for animals or humans, or a structure and chemical pathway such as for flower bud formation. As a very basic level, a reductionist might contend this is a linking of many chemical pathways in response to environment. A wholistic approach, as a preferred approach, realizes there are relations between many component parts within a natural biological organism that are able to coordinate to display AN. Thus, the overall developed structure is sufficient and equipped to sustain the necessary relations for AN that could not occur if the system were dismantled (Rosen, 1991). AN in a system provides desired task achievement by acting *before* the outcome of the situation is certain, so the system appears to know the future. The structure and inner workings of the system and system dynamics all manifest the best behavior choice for existing niche conditions. In biological systems, for choice of behavior, an organism may need to practice to develop the quick response to the perceived conditions. The system designer builds

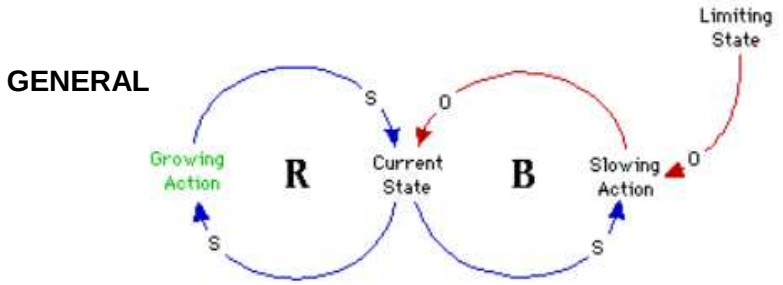
into the system the means to manifest that behavior. For a robotic system, the designer identifies the most likely conditions to be encountered, and the direct matched response is tied to that by a means to detect fitness of the situation, considering multiple factors summed to determine a cue value, and that manifests the behavior by the robotic agent. Several indications are observed to identify that AN is indeed at work (Nadin, 2002). The current behavior is actually determined by an expected future condition. The system contains models of itself and the niche environment. A correlation is made between the perceived niche and resulting behavior, even if existing perceptions are minimal. As a result, the artificial robot organism appears to have a connectedness with the real world. There is apparent synchronization of activity and integration within a dynamic system. Though AN realizes a single behavior choice at a time, it appears to know the future outcome that is best for the manifested behavior. When these elements are observed for any system, whether natural biological ones, or artificial robotic ones, the system can be said to have anticipation. The most important elements to observe are that the system manifests behavior *before* an outcome is certain, so that the observed behavior *appears to have known the future* outcome ahead of time.

ANTICIPATION ARCHETYPES

Although there are around 10 difference business system archetypes, the two that apply most directly to robotics are the Limits To Growth (LTG) archetype, having a reinforcing and a balancing loop with a delay implied in the balancing loop, and the Shifting The Burden (or Goals) (STB) archetype with two balancing and a reinforcing loop. As the name implies, a system showing the LTG contains some constraint or limited factor that directs the system to some asymptotic stability, and this is discussed immediately below. Slightly less clear, the STB

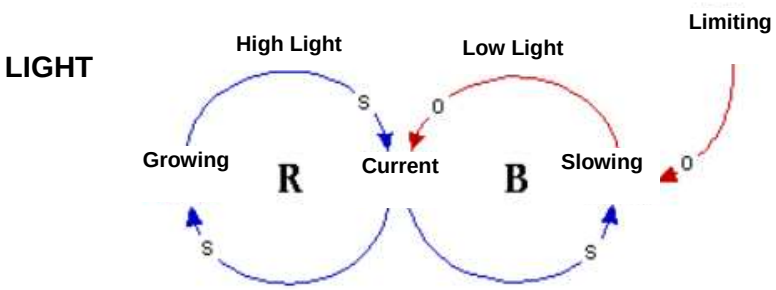
archetype includes a symptomatic short-term solution loop pathway that is easy to implement, but does not solve the problem, while the fundamental solution loop pathway is more difficult to implement, yet leads to a more desirable long-term solution. The STB archetype is presented following the LTG archetype.

Causal loop diagrams represent the LTG archetype (Figs. 21 & 22).



Limits to Growth (Limited Success) (Original)

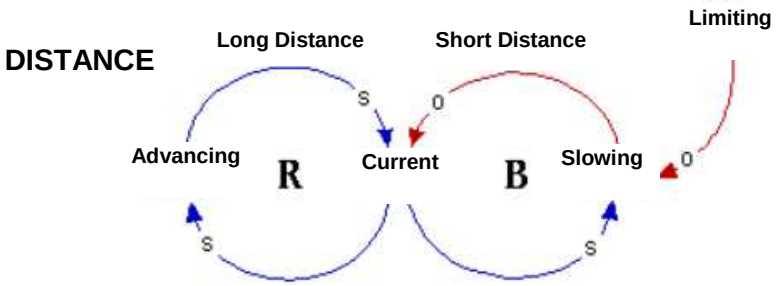
S=Self-enhancing; O=Opposing Balancing



Limits to Growth Due to Light Level

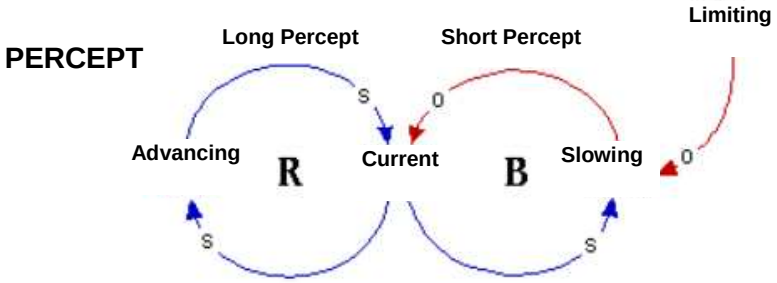
S=Self-enhancing; O=Opposing Balancing

Figure 21. Limit To Growth causal loop diagrams for describing a general situation or response to a light level.



Limit to Growth Due to Robot Distance from Object

S=Self-enhancing; O=Opposing Balancing



Limit to Growth Due to Robot Percept

S=Self-enhancing; O=Opposing Balancing

Figure 22. Limit To Growth causal loop diagrams for describing distance traveled or percept effects in the niche environment.

The LTG archetype is a kind of tug of war between two opposing factors, one that increases response in the reinforcing loop (shown on the left side as R) and reduces response in the balancing loop (right side shown as B). Some limiting element is working on the balancing side to create a constraint so the overall action is to move to some asymptotic value for a stable equilibrium state. Although this may be a desired effect, most likely the balancing loop with the limiting element is some undesired effect that limits the growth or expansion of the system. In business or biology, and likely in robotics, the constraint might be altered to increase the asymptotic performance value without creating an unstable condition.

The LTG archetype is a first order system dynamics model that can be represented in some detail by a stocks and flows map (Fig. 23).

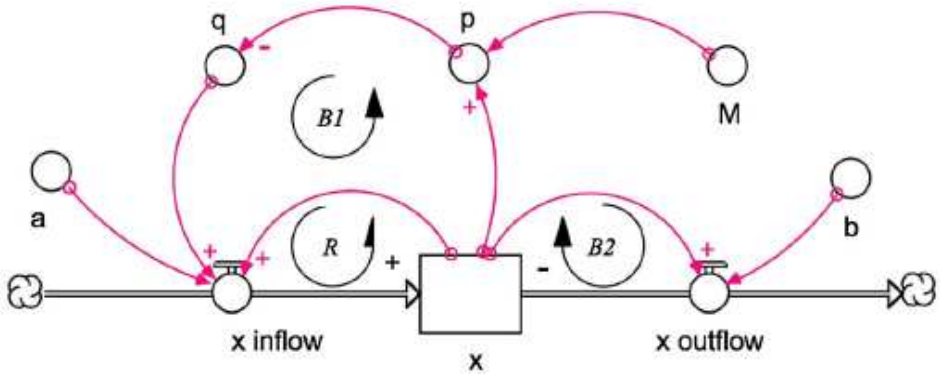


Figure 23. Simulation stocks & flows map for archetype Limits to Growth as a first order model (one variable only: x) (from: Hayward and Boswell, 2014, Fig. 2, related eq. 6).

Differential equations represent the process for the change in x (derivative with time):

$$dx/dt = \dot{x} = ax\,(1 - x/M) - bx \qquad (3.1)$$

and

$$x(t) = x(t\text{-}1) + \dot{x} \quad \text{(Euler integration)} \qquad (3.2)$$

Recall the definition (Sterman, 2000, p. 194): d(Stock)/dt= ΔStock= Inflow(t)-Outflow(t)

and thus: $\text{Stock}(t) = \text{Stock}(t\text{-}1) + \int_{t-1}^{t} [\text{Inflow}(s) - \text{Outflow}(s)]\, ds$ (3.3)

The LTG archetype is thus a single differential equation (3.1) that is integrated, and can be easily used in a computer program with the form above. The LTG archetype version shown here combines one reinforcing (R) and two balancing (B1 & B2) loops, one having the limiting element. The interaction between the various values of the parameters (a, p, q, and M) create the system dynamics. Again, this may constrain the system, or altered values may allow stable equilibrium to be reached at some higher level.

The LTG archetype can be rewritten to be described for a robotic scenario that uses a percept in the niche and resulting behavior (Fig. 24).

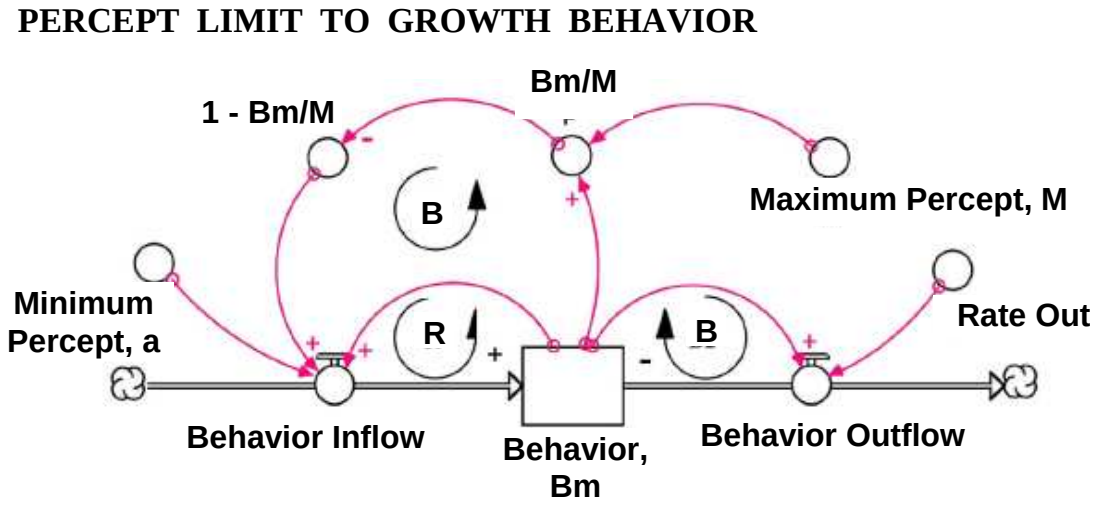


Figure 24. Simulation stocks & flows map for archetype Limits to Growth (Success) for robot distance from an object as a first order model (one variable only: x) (modified from: Hayward and Boswell, 2014, Fig. 2, related eq. 6).

The relations between the values of the factors is shown here for more detailed clarification. Inflows to behavior values and outflows represent the changes in behavior that are observed for the system.

Details of the LTG archetype were organized into a Simulink model to illustrate the system dynamics (Fig. 25) or for an individual percept (Fig. 26).

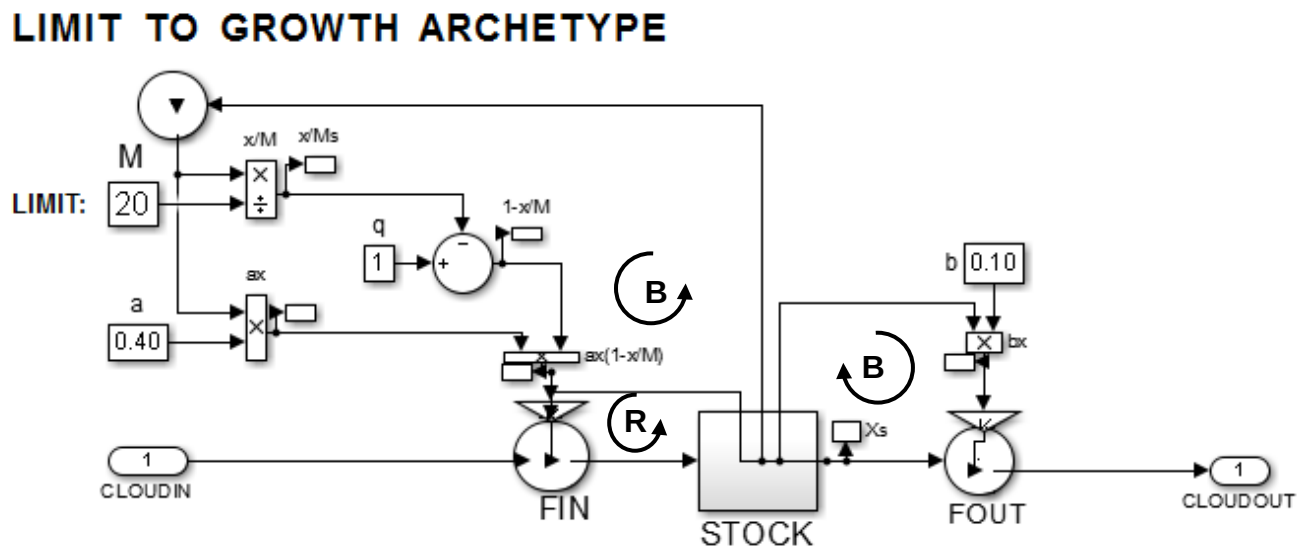


Figure 25. Limit to Growth archetype stocks and flows (levels and rates) map.

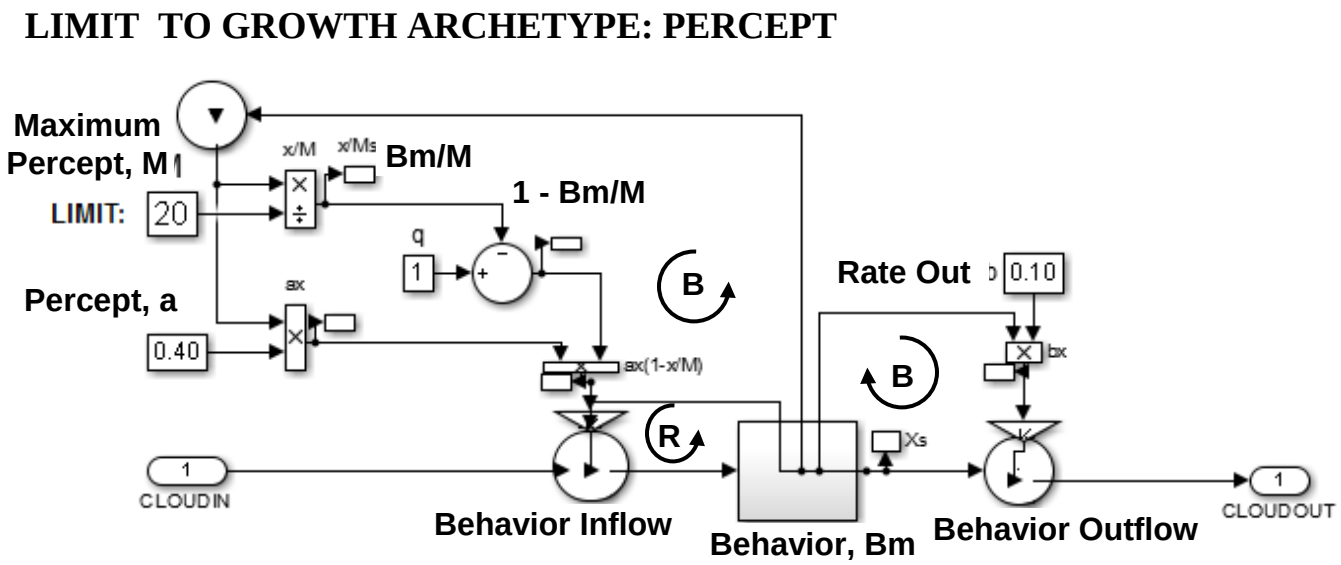


Figure 26. Limit to Growth (Success) for Percept archetype stocks and flows (levels and rates) map with resulting Behavior.

System dynamics for the Limit to Growth archetype as presented by Hayward & Boswell (2014) have been recreated in simulation for the same parameter set of a=0.4, b=0.1, xo=1.0 as a starting value, and a limit of M=20 (Fig. 27). Note the convergence to output growth value of Xs=15 at about time t=20 to 25 for both simulation models, verifying correct operation for the program of the LTG archetype.

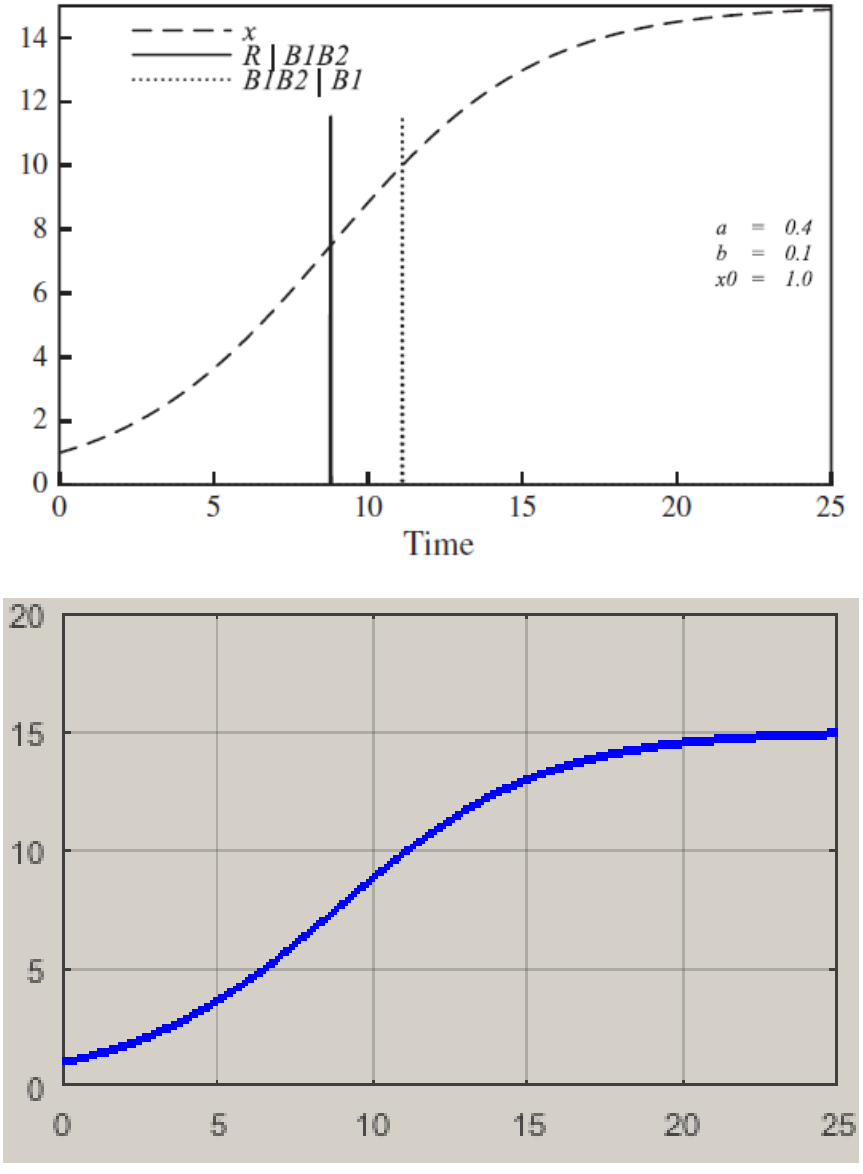


Figure 27. Output for first order loop model for archetype Limit to Growth with loop dominance shown from Hayward & Boswell (2014) (top) and the recreated archetype with similar resulting output (bottom) (From, Hayward & Boswell, 2014, Fig. 3, p. 34).

The *original* shifting the burden (STB) or goal archetype implies that problems arise from the actions that only solve symptoms (Senge, 2006, p. 391), ansd was represented in detail by Dowling et al. (1995) (Fig. 28). A solution only works in the short-term with some immediate positive results, but in the long-term is not sufficient. The ineffective attempted solution is repeated at the expense of an actual fundamental solution that would work in the long-term. Unfortunately, eventually the actual fundamental soultion may no longer work either, as the symptomatic solution is blindly and repeatedly applied. Example: selling more to existing customers rather than increasing market share by selling to new customers.

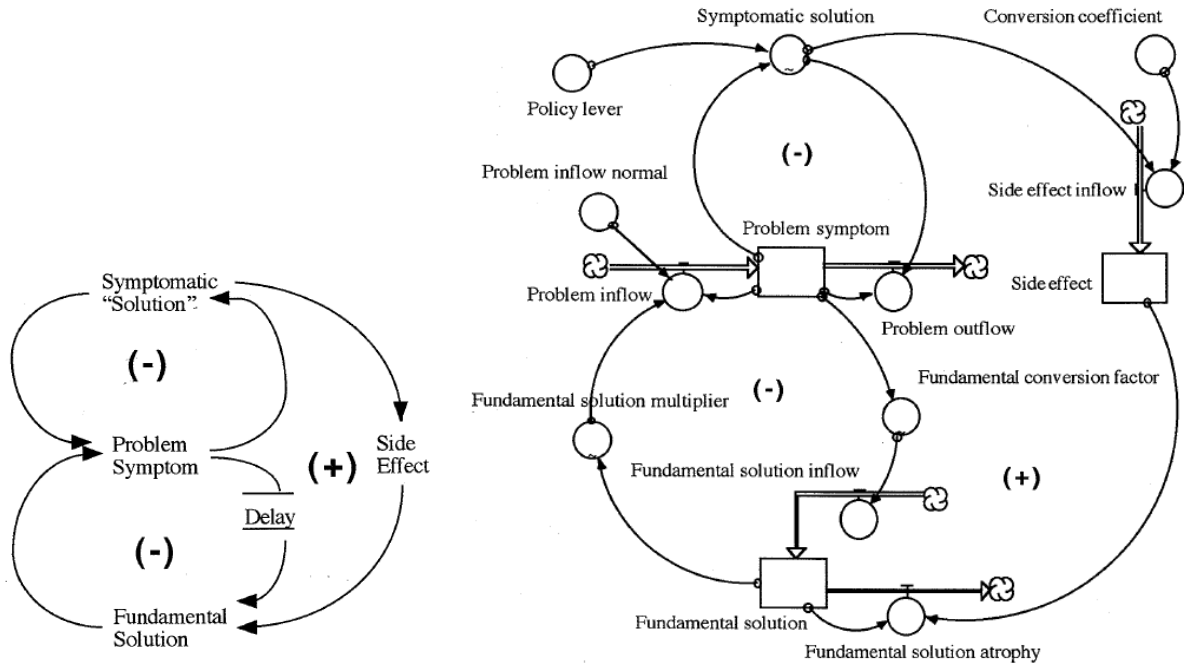


Figure 28. Shifting the Burden archetype causal loop diagram (left) & stock & flow map (right) evidencing the degree of detail added to clarify beyond the causal loop diagram. The system contains two stocks as a second order system, and one pipeline delay (outflow=inflow after delay time) shown as a separate stock. Two negative feedback balancing loops and one dominant positive reinforcing loop interact to create the system dynamics (From: Dowling et al., 1995, Figs. 2A&B, p. 458&459).

Although it sounds like a fit, shifing of goals (or burden) does *not* mean to make dynamic changes in a system model to find creative new solutions matched to the context observed, as

would contrast with that originally presented by Senge (2006). Rather, the ‘shifting the burden (or goal)’ archetype points to the fact that unproductive behavior tends to arise in the long-term from some behavior that seemed to be working in the short-term, while it actually is not, but just treats symptoms, while not treating the core problem.

The STB archetype can be described for a specific situation of path behavior with an altered stock & flow map (Fig. 29). Causal relations are represented by the interactions with Initial Turn (Symptomatic Solution), Distance (Symptom), and creation of a Sussessful Heading (Fundamental Solution), based on a Turn Delay and the limiting side effect of a Late Turn. These are represented in more detail in the stock & flow map (Fig. 29, right).

PERCEPT: PATH BEHAVIOR

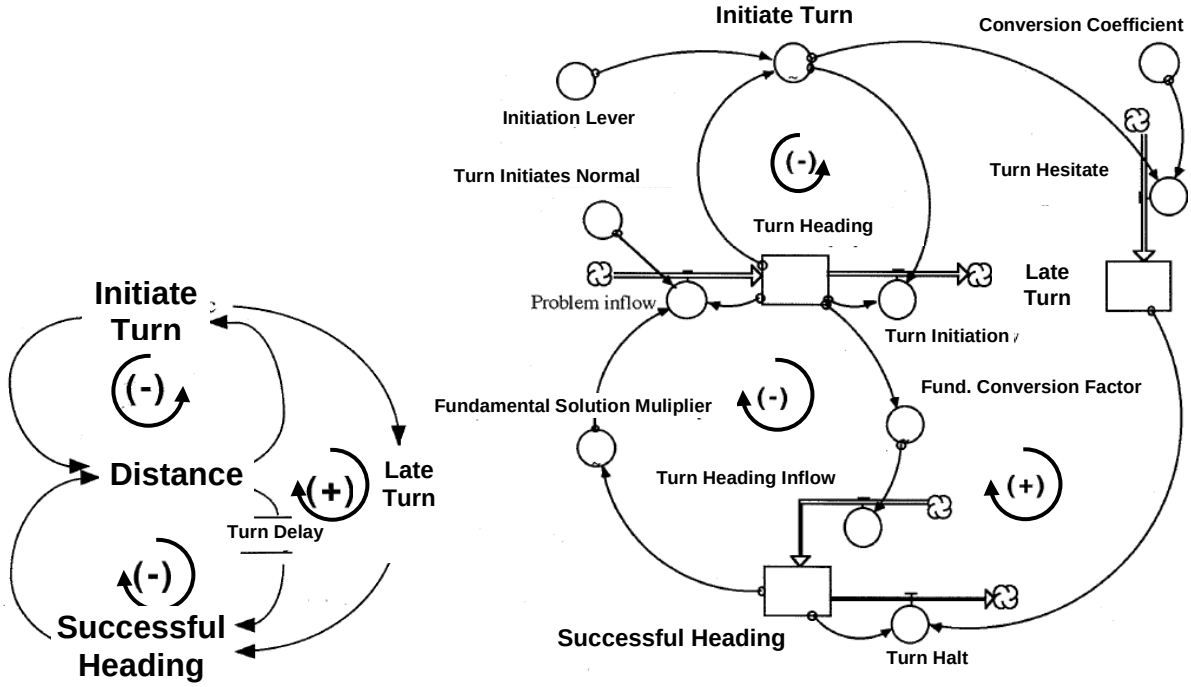


Figure 29. Altering the Shifting The Burden archetype to describe Path Behavior.
Shifting The Burden archetype causal loop diagram (left) & stock & flow map (right) evidencing the degree of detail added to clarify beyond the causal loop diagram. The system contains two stocks as a second order system, and one pipeline delay (outflow=inflow after delay time) shown as a separate stock. Two negative feedback balancing loops and one dominant positive reinforcing loop interact to create the system dynamics (Modified From: Dowling et al., 1995, Figs. 2A&B, p. 458&459).

The STB archetype was represented in Simulink based on work by Dowlinget al. (1995), in attempt to reproduce similar system response (Fig. 30). Overall performance of the previous work was captured and agreed well when the results when the policy lever was set to zero (PL=0, or policy off). However, when the policy lever was turned on (PL=1), the response became unstable in comparison with the previous work. It was possible to change parameters to match the response shown in the previous work, but that showed inconsistencies as the no policy lever setting. Though program code was presented in the previous work, some missing details are

thought to result in this lack of agreement. Thus, a similar STB archetype was developed, yet with some differences in response.

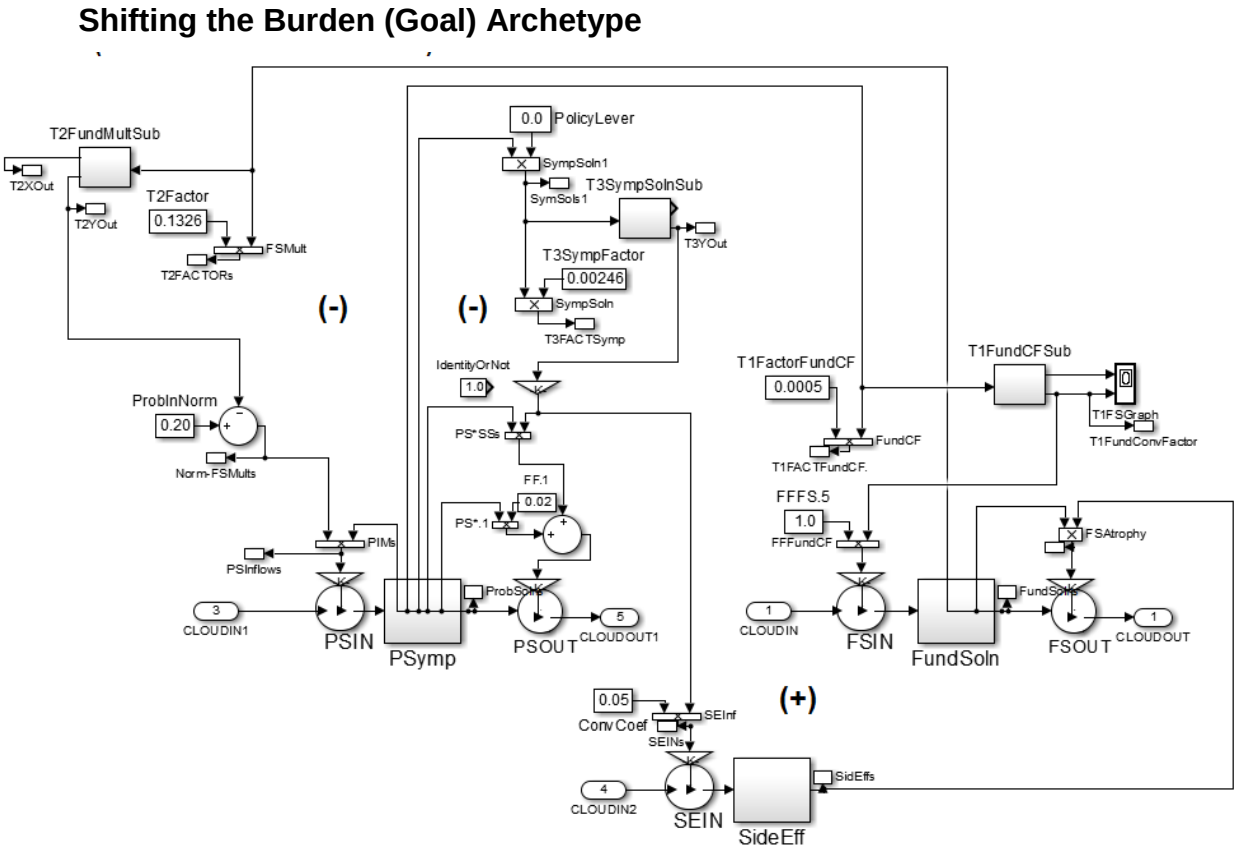


Figure 30. Shift The Burden (Goal) archetype for comparison with Dowling et al., (1995, Fig. C&D). Problem Symptom (PSymp, left) represents short-term solutions that may create a Side Effect (SideEff, bottom), while the preferred long-term Fundamental Solution (FundSoln, right) takes over when the Policy Lever is turned on (PL=0.0 or 1.0 as off or on) (from: Dowling et al., 1995).

Shifting the Burden archetype game

A system dynamics business game was developed and tested (Bagodi and Mahanty, 2015) for a Shifting the Burden archetype architecture, and showed symptomatic solutions were chosen by participant managers instead of the preferred fundamental solution. Apparently the system is too

complex to recognize both the underlying system archetype and a necessary preferred fundamental solution. Perceived improvements in the short run (more hiring and more sales) failed in the long run, since fundamental solution is needed (improved quality) No participant managers could either recognize the STB archetype or find the preferred fundamental solution, and so failed in performance. The underlying causal structure was revised to be representative of a robotic architecture (Fig. 31). Also, the flow map was revised as a potential underlying structure for the STB archetype both in a biological plant (bioplant) growth system and for a mobile robot system (Figs. 32 & 33). This illustrates how a generalized STB archetype structure can be revised to conform to similar homologous systems. The first is a bioplant growth system with shifts to flowering based on photoperiod (daylength). The second is a mobile robotic system that changes heading to match percepts from the niche environment. Thus, aspects of diverse yet homologous systems can be represented with the STB archetype.

PERCEPT: PATH BEHAVIOR

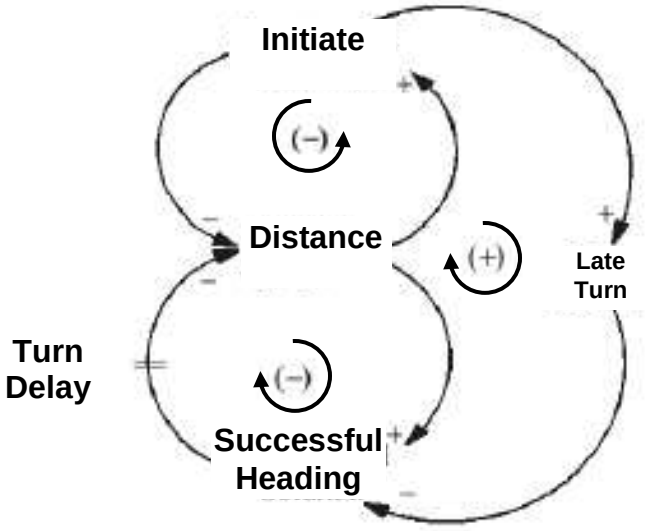


Figure 31. Underlying causes for Shifting the Burden archetype for robot path behavior (after: Bagodi and Mahanty, 2015, pp. 387, Fig. 2).

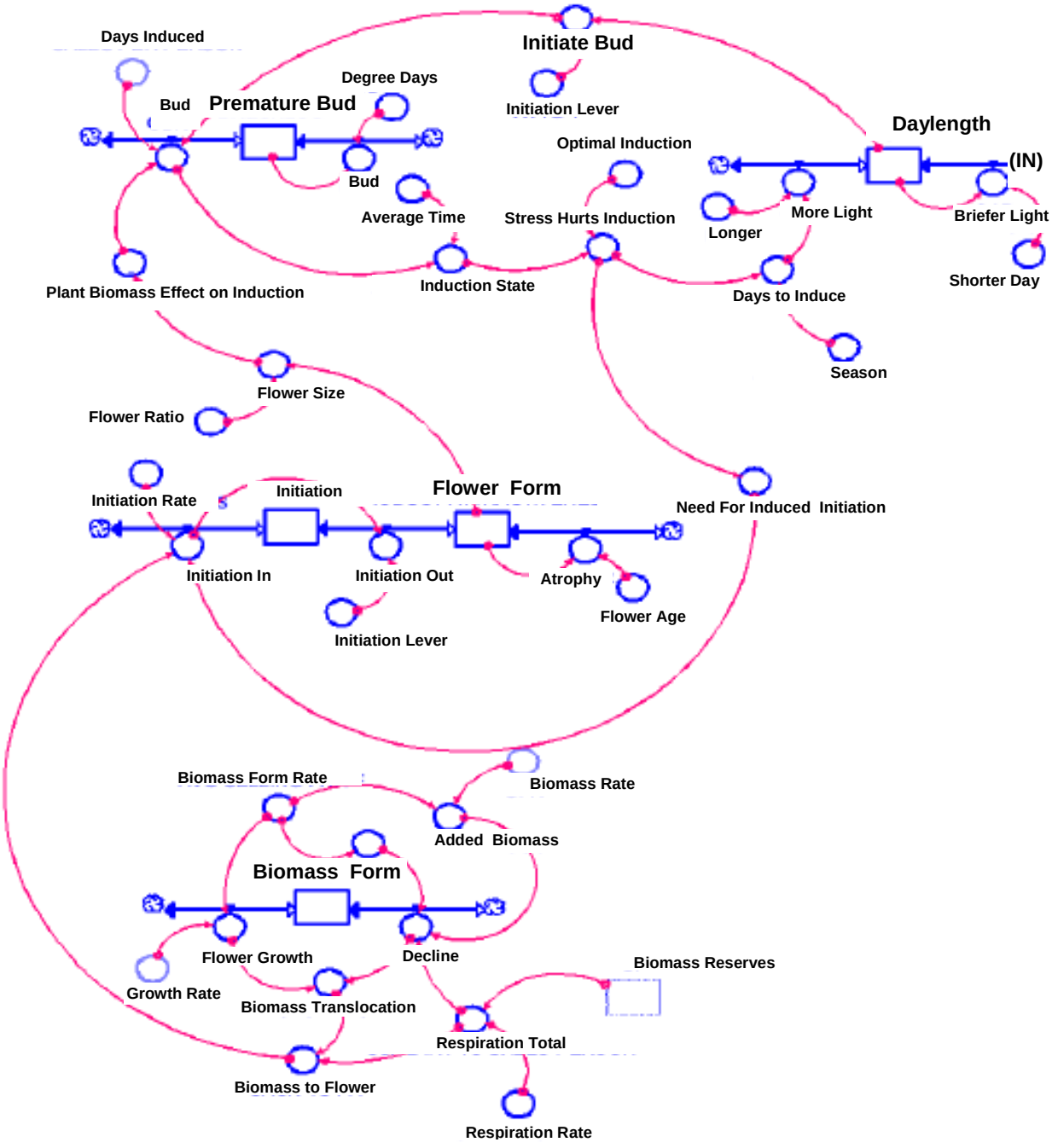


Figure 32. Flow diagram of the simulation model for the Shifting the Burden archetype business system game, revised to reflect photoperiod affects on buds and flowers (after: Bagodi and Mahanty, 2015, p. 389, Fig. 3).

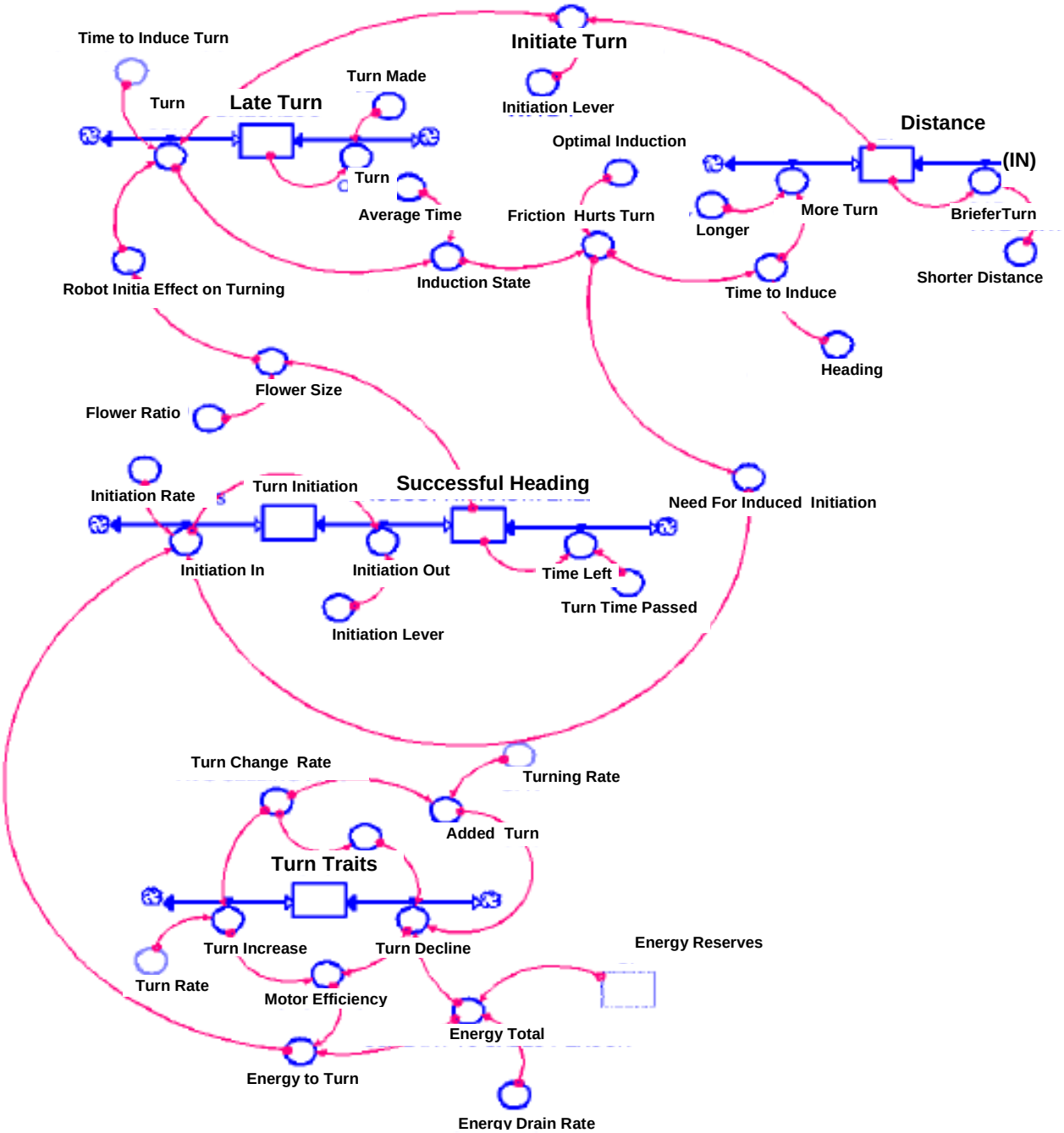
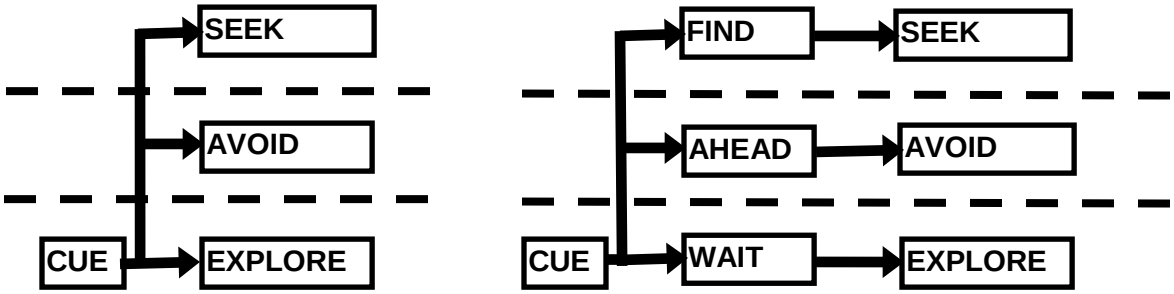


Figure 33. Flow diagram of the simulation model for the Shifting the Burden archetype business system game, revised for percept effect on robot behavior path by turn and heading change (after: Bagodi and Mahanty, 2015, p. 389, Fig. 3).

ANTICIPATION IN ARCHITECTURE

Anticipation (AN) goes beyond mere reactive behavior normally used in behavior-based robotics (BBR). As with BBR, the initial defined levels can be built upon in a vertical structure. Simple behaviors used by the wall follower (WF), or MURAMATOR for 'wall lover' in Latin, are EXPLORE, AVOID, and SEEK. A simple addition of AN to the WF to create a TOURIST can be achieved by addition of just three additional behaviors: WAIT, AHEAD, and FIND (Fig. 34). A basic WAIT behavior procedure is included at the start to allow all systems to come up to speed, and can be cued whenever an uncertain situation is encountered, so the robot does not appear to panic with an illogical behavior in response to an unexpected, and thus unanticipated, situation.



WALL FOLLOWER **TOURIST**

Figure 34. Representation of behavior choices in the wall follower robot (or MURAMATOR) for three basic behaviors (left) and addition of three behaviors to the TOURIST robot (right) to include anticipation for task achievement of finding objects and following walls.

AHEAD is included to extend beyond mere reaction to a wall at a specific distance (a cue for AVOID), and thus includes an extended gradual response to presence of an ongoing wall object rather than an abrupt direction change at some critical distance, and possibly unsafe approach

distance reached. Lastly, FIND is added in relation to the SEEK behavior procedure to search out potential objects and walls nearby during the SEEK spin motion, and thus may easily locate a more acceptable object to approach and follow along as a wall.

MODELING ANTICIPATION SYSTEMS

Architecture for anticipation has a linkage between the niche environment and the observed behavior choice (Fig. 35).

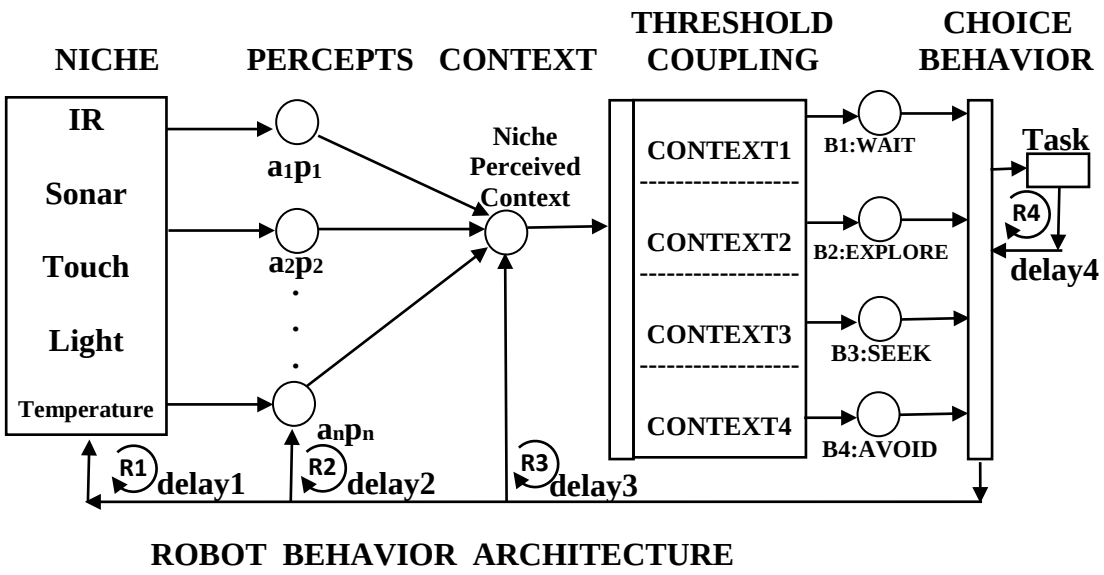


Figure 35. Architecture for robotics relating perceived environmental niche to context for robot behavior with reinforcing loop back to the niche [causal diagram and flow map].

The causal diagram and flow map above was provided with more detail and specific coded processes to represent the underlying routines and looping structure, and delays to represent the notion of anticipation (Fig. 36)

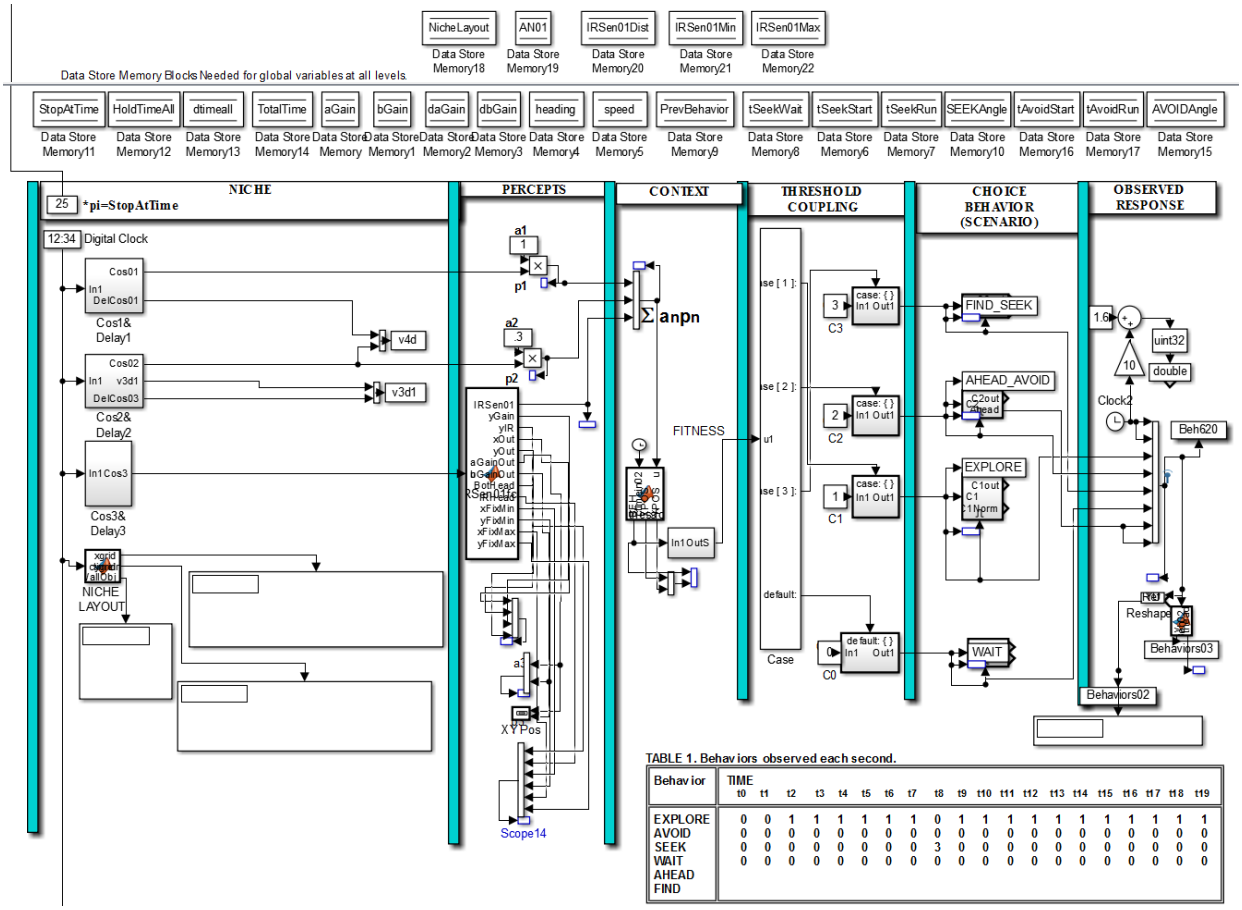


Figure 36. Anticipation simulation model for TOURIST robot agent. NICHE creates a 100 X 100 pixel arena within which the agent perceives the conditions (PERCEPTS) and from that determines a CONTEXT fitness value that is used through THRESHOLD COUPLING to manifest a CHOICE BEHAVIOR that matches the current Niche conditions, thereby producing the OBSERVED RESPONSE that leads to successful task achievement. Anticipation can be turned off or on as desired.

A niche is a specific environmental surrounding perceivable by the agent. It can be considered as a smaller subset of the overall environment or arena that the robot agent is able to exist within, and is a local condition or context that is relevant to manifestation of a behavior choice (Fig. 37). As implied by Simon (1996), items that are too far removed in space and/or time from an organism (or robot agent) should not be considered relevant to the problem at hand, and thus the behavior choice. Indeed, the term environment generally is used broadly to indicate almost any area that an organism (living or a robotic agent) can move in and exist within, and may be

considered as broad as the entire planet earth, especially when we consider broad effects to the environment such as global climate change. However, on a realistic scale the location and typical movement of an individual within the larger environment is quite readily refined to a smaller localized area, one considered as home, or a slightly larger territory. Therefore, the operation environment for an organism or agent is curtailed to one that to which it is functionally matched. The immediate area influencing behavior choice is smaller still, defined by the local area in which significant effects are perceived, and thus manifest some behavior.

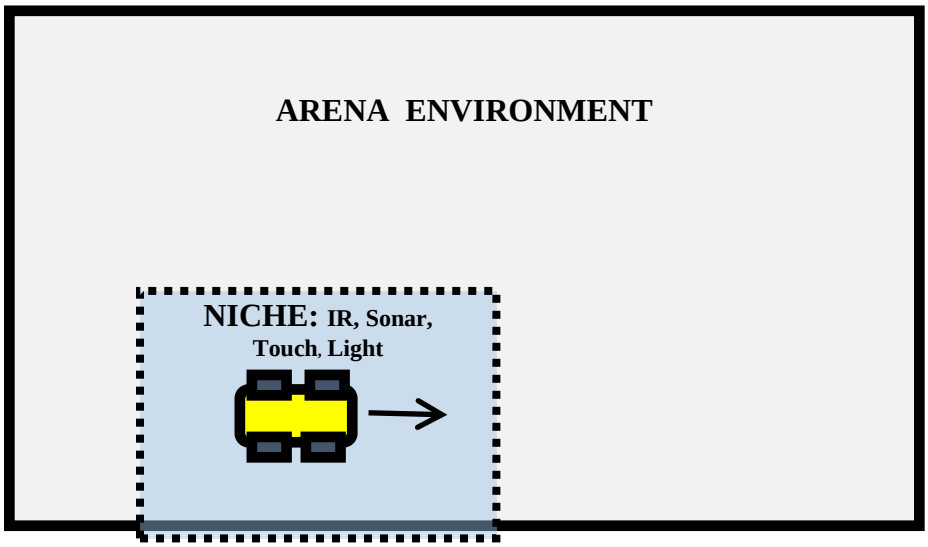


Figure 37. A niche is shown as a specific environmental surrounding perceivable by the agent, and is in the immediate vicinity as a local condition or context.

Percepts are the current perceived conditions in the niche. They are used in combination to determine a suitable fitness level. That suitable fitness is matched to a behavior that is manifest in response to the niche conditions. A sequence of behavior choices based on the threshold levels leads to desired task achievement as an observed response.

CONGRUENCE FRAMEWORK

A congruence framework for modeling a real world system ensures that the causality is first abstracted by encoding into a formal system (FS) of equation relations and rules, and the inference captured in the FS is decoded back to the natural system (NS) of the real world (Fig. 38). The framework is said to be congruent, or agrees, if the entailments (implications) in the causality of the original NS can be reproduced when the inference of the FS is decoded back to the NS (Rosen, 1991).

Rosen terms *metaphor* the process of decoding without encoding, or forming a theoretical model of inferences without initial measurements (p. 65), perhaps by only the application of first principles of physics or other disciplines (and he says this is not science, p. 66). He refers to the efforts of von Bertalanffy in the 1930's or later (or von Bertanffy, 1968) drawing metaphors between open systems and biological development in morphogenesis (p. 65). The result is a belief or expectation that a metaphor can be a useful model to be decoded into some biological open system, thereby establishing a modeling relationship to better understand the biological system (p. 65). Unfortunately, skipping the encoding loses precise verifiability. Yet, the metaphor can embody truth, yielding some type of gain for free, though it is unverified (p. 66).

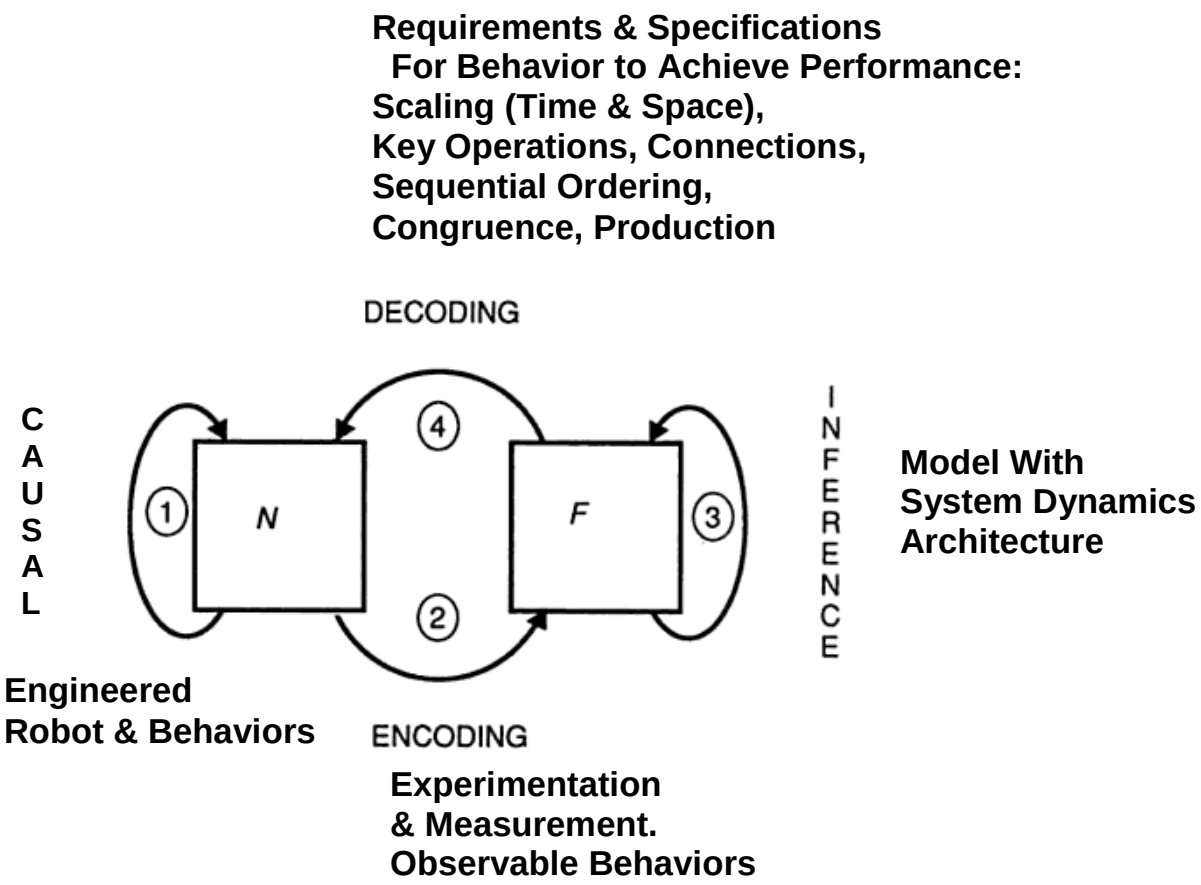


Figure 38. Rosen proposed the need for encoding and decoding to link causation relations between world phenomena into an embodied model structure. A Natural System (N, or later referred to in this discussion as NS) can be modeled by a Formal System (F, or herein FS) by adding processes of encoding and decoding as creative acts. The circled labeled paths are related by the equivalence: 1 = 2 plus 3 plus 4, or meaning that path 1 is equivalent to the combination of the other three paths. (from: Rosen, 1991, p. 60; Fig. 3H.2)

Since the modeling relation includes the notion of prediction, the future is implied (or entailed) by the present in the causality of the NS (Rosen, 1991, p. 64). The individual must decide a course to provide the proper encoding and decodings. It is imperative to discover the right encodings (and decodings). As the metaphor provides the decoding without encoding, the advantage is to decode to the NS from some devised FS without bothering to encode the particulars known about the NS. All of science, and biology specifically, uses such metaphors, so that metaphors about machines can be used to decode predictions about organisms (p. 65). Thus, Rosen explains away the need for encoding in his original model of abstraction, though this seems less than logical.

Yet from Rosen (1991) there would need to be a means of decoding, though it is not specific, but is necessary for the individual to perform as a creative act (p. 54). Many means may be possible to achieve this. Several logical processes would seem needed to be included in a useful method, and those can be identified as: scaling in both space and time, key operations, forming connections, sequential ordering, comparison of options, and design for manufacturing and assembly.

Scaling of the system in both *space and time* is crucial to identify means of producing some desired task achievement. Taking a point of view that zooms out (reduces to a bird's-eye view) or zooms in (enlarges) for a factor of 10 times (or even 100 times), either with technology or at least with ones own imagination, should uncover elements that can be exploited to achieve a task. By comparing with a mental view, by describing the new point of view, or listing items in a table one can discover intricacies basic to the solution of the problem. These items and relations

must be captured in the FS, or additional mathematical equation relations devised to represent any new found concerns. Most importantly, one is less likely to overlook any items of importance that can be enlisted to achieve the task. Scaling in time can range from nanoseconds (for speed of program execution or biological chemical reactions) to decades or more (product life cycle, reuse, recycling, or expected appropriateness of technology). For a robot, the turning mechanism would be considered for a distant view, or a detail of sensing may be improved for an enlarged view. Time of processing should be fast enough to sense and interpret percepts in real time, while moving around fast enough in an arena to achieve a task in a reasonable expected time frame. Einstein (1905) had used this stance in his quest to form the theory of relativity, as he explained his ideas came at least in part from imagining riding on a beam of light, requiring scaling in both space and time.

Key operations must be described as relations among parts so that desired steps in operation can be accomplished toward the goal. This can be done both in a normal view, and in the scaled views already mentioned. This identifies key parts and the features of each to be altered and possibly optimized, or in contrast generalized, so a proper description made mathematically in the FS can be assured to be translated into a feasible NS. Size and shape of individual parts, as well as strength and compliance properties, should be transferred to the NS from a FS. A mobile robot would need proper wheel size with angular velocity and torque such as needed to navigate in a desired arena.

Forming connections that are robust and yet tenable for the prescribed operation are a more detailed manner in which the operations must be related and carried out. Connections between

various component parts must be strong yet able to provide needed movement. Many times failure is known to occur at connections, either from poor initial forming of the connection, or mistaken consideration of how the system operation will add stress or undue forces at the connection. The nature of the connection likely is critical to proper operation of the system, and alternative forms of connection can drastically change performance. The interactions that result from the connection are key to producing the desired behavior. Recall that forming connections is one means of self-organization (Ashby discussed by von Bertalanffy, 1968, p. 96) that requires an open system, decrease in entropy inside the system, and both inputs and outputs from/to outside the system with the environment. The decrease in entropy is counter to the Second Law of Thermodynamics, and would likely naturally lead to breaking such connections for increased entropy with resulting failures during system operation. A mobile robot needs axle connections to both a motor and the wheels to provide aligned motion of all wheels to make for ease of movement without pinching or obstruction.

Sequential ordering should assure all needed items are included, and can proceed in time to lead to task achievement. Parts, connections, and their interactions must perform in a proper sequence to add up to the desired result. Invariants in the environment cue the sequence for successful operation. These cues are provided both internally to the system and externally from the environmental niche. Altering the order and timing leads to degraded or lack of performance. A mobile robot must have perception of the niche in a way that avoids striking objects in its direct path, and must look for objects that it would anticipate to encounter in the niche.

Comparison of options that arise during the previous formulations provides a means by which to select those individual steps that are considered most successful, and ensure the combination is one that produces the most elegant successful result. Comparisons can be made on desirability of operation, cost, feasibility of production, or efficiency of operation. For a mobile robot, desirable operation is preferred as observed operation in the real physical world.

And in final production, *design for manufacturing and assembly* instills in the solution the means of engineering that is most likely to make a useful item that also is economical to create (Boothroyd, 1994). Creation of simple pieces, and the fewest number of pieces, favors both manufacture and assembly. Design is an iterative process that considers both design knowledge (Kdn) and domain knowledge (Kdm) while considering specifications and requirements for construction (Troxell and Troxell, 2014). Requirements include both criteria and constraints. Criteria are desired traits for successful structure (e.g., functional operation or efficiency), while constraints are limitations to producing the solution (e.g., funding or size) (https://www.nagb.org/publications/frameworks/technology/2014-technology-framework/toc/ch_2/design/design2.html). A minimalist view for a mobile robot uses the fewest motors and wheels needed, while using a minimum of microprocessing power, especially by building the robot with structures that are well-matched to the environmental niche. Indeed, a smaller number of simply designed parts is much easier to assemble. Having fewer parts requires less fasteners to connect them together, thereby greatly reducing time, labor, and resources for the proper production of the final product. Overall, consideration of scaling, identifying key operation, connections, sequencing, comparing all options, and design for assembly lead to a more effective design process.

INSTANCE OF ANTICIPATION

A simple mobile robot was designed and built according to a minimalist approach philosophy to permit study of the basic notion of anticipation. A foundation structure and operation was followed from Connell (1990) who made a wall following robot (termed MURAMATOR, meaning “wall lover” in Latin) using subsumption architecture of Brooks (1999) developed for behavior-based robotics. It included only three behaviors: EXPLORE, AVOID, and SEEK. A single infrared (IR) sensor was mounted at a slight angle from straight forward to detect any object in the path of forward travel, with no other sensing capability added. Its realm of knowing was only briefly in front (slightly angled) and the open area just traveled behind it. When some object is detected, the simple AVOID behavior moves backward briefly and turns by locking a ratchet mechanism in the right wheel to make a left turn in place. When EXPLORE continues for some designated amount of time without detecting an object (or wall), the simple SEEK behavior cues to make an uninterrupted 270 degree counterclockwise (CCW) turn to the left (by adjusting a voltage trimpot to generally time the spin motion), which is effectively a 90 degree clockwise (CW) turn to the right. The overall outcome of these three behaviors working together is to produce **task achievement** that is twofold: 1) **follow along a wall** in reaction to detecting the wall, and 2) **avoid any stationary objects** that are encountered separate from the wall. **Note that the three behaviors (EXPLORE, AVOID, and SEEK) each by themselves do not achieve the task, but the interrelation of the three individual behaviors result in an overall behavior result that is the task achievement of wall following and object avoidance.** Thus, a principled approach with rules of operation for coordination of the behaviors results in task achievement.

Initially the TOURIST robot was studied in an arena as a natural system (NS) to confirm and validate function, and the processing program on a microprocessor (i.e., Arduino Nano) was encoded into a simulation program for further study. The name TOURIST was coined to reflect the robot agent tendencies to look ahead to search out objects to find, encounter, avoid and follow when subsequently approached close enough. SIMULINK was used as a graphic virtual programming language to encode the desired minimalist anticipation architecture formal system framework, called ANSIM (for ANticipation SIMulation). SIMULINK provides benefits for a useful graphic display of programming architecture with structured framework for underlying coding, and allows for additional features to be coded at lower levels using MATLAB subsystem modules that can carry out the details of the operations, similar to subcognitive modules enlisted by Malcom and Smithers (1990). Archetypes of Limits To Growth (LTG) and Shifting The Burden (STB) (Senge, 2006) were first developed after existing coded examples of the archetypes functions (Hayward and Boswell, 2014; Dowling et al., 2006) so that correct operation was verified, and the essential component code was added into the ANSIM model architecture to enable addition of the notion of anticipation with system dynamics as traits of the archetypes. First, modifications were initially made iteratively to the ANSIM FS model to attain desired behaviors for the simple wall follower robot agent (i.e, reasonably short enough response time, and proper AVOID response to arena boundaries and objects). The three basic behaviors of Connell (1990) as EXPLORE, AVOID, and SEEK were all shown to work correctly in combinations possible to exist in the arena and with up to two stationary objects present. This basic framework was then used to add the notion of anticipation with additions of the archetype programming to the ANSIM FS.

Archetype operation for the notion of anticipation was added to the ANSIM FS model in three specific ways. First, the WAIT behavior was added at the beginning of the robot startup initialization to allow for equilibrium to be achieved internally before any response to the niche was taken. WAIT might also be used later during runtime when an unknown condition (e.g.. unexpected and unaccounted for) is encountered, and thus keep from performing any illogical or unreasonable behavior with potentially destructive or injurious results to the robot. Thus, potential injurious results were anticipated and a behavior in place to respond reasonably. A second more active AHEAD behavior, using the LTG archetype, was added previous to the existing AVOID behavior module to scale reduction of the time of turning for AVOID in relation to an object at a distance, one that has yet to be encountered for needed avoidance, but still in need of response to in an expected future. A third FIND behavior, representing the STB archetype, was added to interrupt the 270 degree spin in the SEEK behavior that was previously not interrupted for SEEK with no anticipation. Thus, FIND locates objects (or a wall) in the distance, and halts the SEEK spin to begin EXPLORE, a straight forward movement, to move toward the found object. So unlike the SEEK with no anticipation that attempted to travel along the wall the robot was already next to, the FIND and SEEK behaviors with anticipation turned on may locate another object (or wall) to move toward. If only one wall is near that the TOURIST robot is already following, it still will return to follow that nearby wall. The combination of these three additional behaviors (WAIT, AHEAD, and FIND) with the previous ones of wall following (EXPLORE, AVOID, and SEEK) from Connell (1990) are an anticipation set of a few needed behaviors that add anticipation to the ANSIM model. A simple switch was added to ANSIM to allow anticipation to be left off or turned on, and thus study by comparison the addition of anticipation to the ANSIM operation in the same niche environment.

Various simulations of the inference entailments captured in the code were studied. First, the ability of the TOURIST to follow the arena wall was verified in simulation, and resulting path and behavior over time was graphed to show the overall results to be consistent with expectations for a physical world. This operation was shown both with No Anticipation (NO AN) or with Anticipation On (AN ON). Again the results were displayed over simulated time by showing the resulting path and corresponding behavior choices. The results of these comparisons are shown in detail in the next Chapter 4 for applications of the simulation. Results are shown for both the open arena having only wall boundaries, and for two stationary objects place in the arena. The graphs demonstrate the feasibility of adding anticipation to a robot in simulation using a minimalist approach, and points out potential flaws that actually are typical of even a simple niche environment, and of the operation of the previous simple robot approach of Connell (1990) to follow walls.

Subsequently, following the congruence framework, the abstracted ANSIM inferences in the formal system (FS) were decoded back into the NS of the physical TOURIST robot by making of SIMULINK code into the Processing code used by the robot microprocessor (i.e., Arduino Nano). Trials were conducted to confirm the effects that addition of the methods for incorporating anticipation had on the operation of the physical TOURIST robot. Video records of the trial showed how the operation with No Anticipation (NO AN) or with Anticipation On (AN ON) differed and resulted in observable benefits from anticipation.

ANTICIPATION ENHANCED HABITS

A cycle that includes cue, routine, reward is defined by Duhigg (2014) to be a habit that an organism is conditioned to perform repeatedly. A study with a monkey recognized an increase in neural activity at a surprising point in the sequence: after the cue (display on a computer screen), and before any routine action (alever press) and reward (a juice offering). This activity was termed anticipation, and rightly so because it occurred *before* the routine action that would produce the reward. As for living organisms, a robotic agent can display anticipation by any activity that comes before the certainty of the outcome even if a chain of events is destined to occur that looks like a habit. Another term for habits in natural living organisms is a fixed action pattern (FAP), or a series of actions that follow a cue to carry out a specific task, and end when some target is reached, and thus runs to completion (Hopper, 2008; Tinbergen, 1951). FAPs are thought to be hard-wired and possibly instinct driven. Since they are built in, FAPs act like habits to attain task achievement. The addition of anticipation to either FAPs or habits enhances the execution of the overall sequence (cue, routine, reward), and promotes successful completion of the habitual response. With a focus on robotics, the cue of a routine may have some associated fitness of suitability that enhances the behavior choice to match the cue in the niche condition. Proper definition of a fitness that cues the behavior entails the inference and causality, thus obtaining a congruence of observed behavior in a natural system that one built in to the formal system model. Thus, a component of anticipation is built in to the robot behavior choices, and that can promote a matching behavior to be manifest for the appropriate niche condition, and run a sequence to completion *before* the outcome is certain.

IMPLICATIONS OF ANTICIPATION

Since AN manifests response behavior before an actual outcome is certain, the response has potential to be quicker and more reliable than reactionary response without AN. Operation with AN continuously updates the response to the perceived niche, and can make adjustments for perceived changes in the niche that cue a change in the behavior. Small adjustments over time use less energy in a timely manner to stay on track with desired behavior choices, so ultimately use less energy over time than would occur for reactionary approaches that use large amounts of energy to make bigger more abrupt adjustments. Though some AN responses may make abrupt adjustments when needed, those occur less frequently, and only when the niche changes abruptly (e.g., at corners), and thus again use less resources. Unfortunately, the early response enabled by AN may actually create an undesirable response if the actual future condition of the niche does not align with that which was expected. Yet, the continuous update with AN should allow an organism or agent to make some type of adjustment, albeit slightly delayed, that should bring the situation under control again to match the niche condition to a desirable behavior for task achievement. Overall, anticipation should develop a smoother operation than for pure reactions, with quick response time, reliable behavior choice, incremental changes, and resulting response before the outcome is certain that appears to respond in a way that the organism or agent seemed to know the future outcome ahead of time.

CHAPTER 4. APPLICATION AND SIMULATIONS

OVERVIEW

An anticipation simulation (ANSIM) was developed to allow various formal system (FS) conditions to be studied. ANSIM was based initially on working robot microprocessor programming, and was modified to a version that was verified to operate as expected in the virtual FS arena. Two different system dynamic archetypes (Limits To Growth, and Shifting The Burden) were incorporated into the ANSIM model to add the notion of anticipation (AN) to the model operation. The model was able to operate either with or without anticipation turned on to make similar simulation runs that can be compared for response in the virtual arena. Insights from the comparisons were incorporated into a TOURIST robot with operation having the traits of anticipation.

CONGRUENCE SIMULATION MODEL

A congruence model framework was built on the work of Rosen (1991, 2014). Causality of entailments (implications) in a natural system (NS) are encoded into the inference entailments in an abstracted formal system (FS). The rules in the abstracted FS can be studied in simulation to determine likely results expected to be realized in the NS. The rules and equations must be decoded through engineering efforts back to an instance of the NS. When congruence (agreement) is realized between the results in both the FS and NS, then the FS can be termed a model of the NS, and the NS is truly an instance of the abstractions designed into the FS (Fig 39). This framework is used to develop an abstracted FS representing a robotic agent system with anticipation, and decoding methods used to translate those rules back into a physical instance NS that can perform behaviors showing the notion of anticipation.

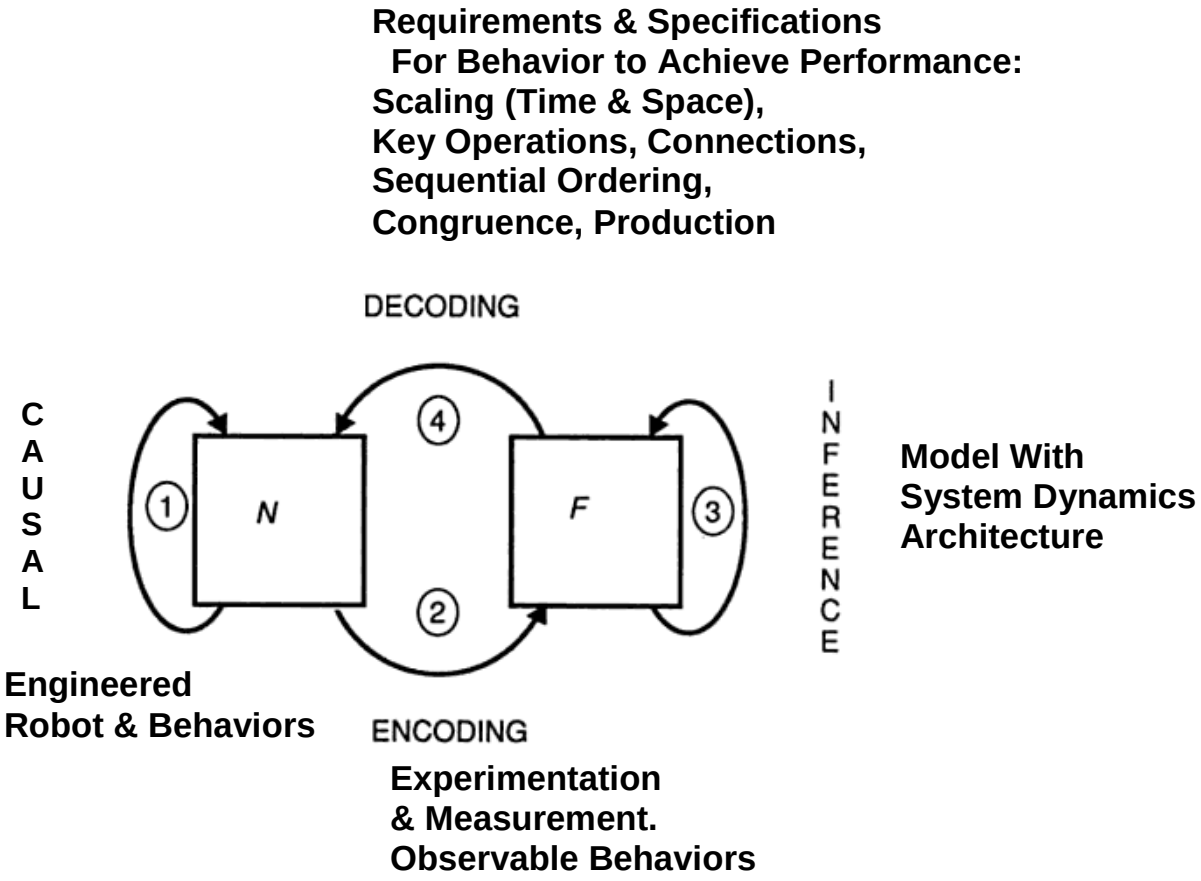


Figure 39. Rosen proposed the need for encoding and decoding to link causation relations between world phenomena into an embodied model structure. A Natural System (N, or later referred to in this discussion as NS) can be modeled by a Formal System (F, or herein FS) by adding processes of encoding and decoding as creative acts. The circled labeled paths are related by the equivalence: 1 = 2 plus 3 plus 4, or meaning that path 1 is equivalent to the combination of the other three paths. (from: Rosen, 1991, p. 60; Fig. 3H.2)

When an NS already exists in the natural world, encoding from the NS to a FS can be derived through experimentation and observation. If actual NS does not yet exist, the idea of *metaphor* is used to create the inference rules of the FS, by using rules about the real world as expressed by laws of physics, chemistry, and mathematics (Rosen 1991, 2014). For robotics, it was possible to

use example program code from a preliminary instance of the TOURIST robot to devise the basics of a simulation in the virtual realm as a base model, termed ANSIM (for ANticipation SIMulation). However, no robot is thought to exist that has the actual notion of anticipation. Instead, the idea of *metaphor* was used to incorporate anticipation into the robot operation. Two archetypes based on system dynamics principles were used to make the rules for behavior response that included a notion of anticipation into the robot FS. Study of various scenarios of anticipation in the FS provided insight as to how anticipation would be observed in a NS when the mathematical rules were decoded back to a modified NS. The rules in program code could be translated from the ANSIM FS model to the Processing program (decoding) in the physical robot microprocessor. This allows for the notion of anticipation that is studied in the FS to be decoded back into a physical robot to operate in a NS.

BIOLOGICAL PARALLELS IN BIOPLANT DEVELOPMENT

Architecture created for the operation of an artificial system such as a robot agent can be applied more generally to a natural biological system for an architecture of the multiple stages for biological plant (bioplant) growth, development, and reproduction (Fig. 40). In both the artificial robot system and the natural bioplant system, a niche environment is perceived to form percepts for the current condition. A combination of the percept factors (infrared or IR, light, temperature, etc.) is used to determine a fitness or suitability for a specific behavior to match the niche

AN in robot systems for AN analogy in bioplant systems.

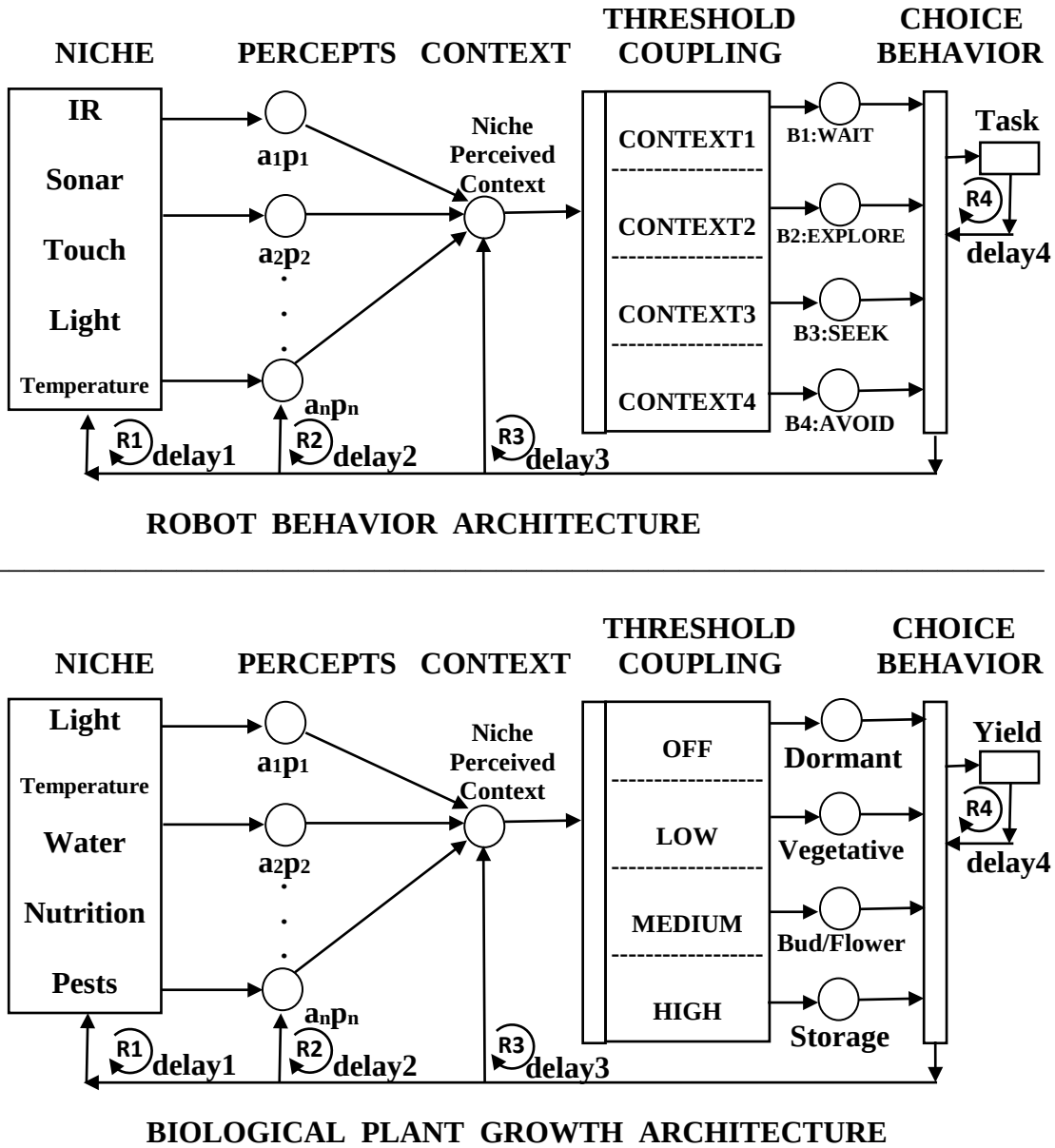


Figure 40. Architecture for robotics relating perceived environmental niche to context for robot behavior with reinforcing loop back to the niche (top) and modified for plant architecture (bottom) [causal diagram and flow map].

condition. Within that niche context, a threshold coupling makes a type of selection to manifest the preferred behavior for that condition. For the robot, it is a mathematical integral of the

multiple niche percept factors that cue a change in behavior. For the bioplant, the integration occurs in biochemical pathways that create threshold levels of chemical molecules that can cue the initiation, and subsequent continuation, of a change in development stage, or nuances within that stage. Hence, a bioplant changes from the vegetative stage (forming only leaves) to a reproductive stage (forming flower buds) based on a combination of percepts of the niche that cue the change.

As with the artificial robot architecture, delays and looping of system dynamics impart a time frame for system response time, or time constants, of the system. Interestingly, bioplant growth is relatively slow compared to that of animal daily activities, so the time constants for bioplants appear to be orders of magnitude slower (longer times) than one would determine for animals (and what we ascribe to robotic systems). Bioplant time constants may be as short as hourly for leaf alignment to light, to a day for circadian rhythms, to months or seasonal for flower buds that develop in the summer to bloom the next spring. Of course, animal and robotic responses are on the order of milliseconds for quick responses, or may be up to months for animal preparation for reproductive cycles and rearing of young, and possibly longer for creation of living quarters to span over decades.

Anticipation of future events is tied up with the time constants for response. Both natural bioplants and animals have inherent physical structures and biochemical pathways that lead to preparation for and subsequent manifesting of behavior choices that lead to attaining desired goals for survival and reproduction.

Evolutionary processes have worked on the ontogeny and phyllogeny of these natural biological living systems (considered open systems in general systems theory) to develop specific behaviors to match a niche, and attain survival outcomes. The structure and operation of a model termed Churchland's crab presents a theoretical discussion of such principles (Churchland, 1986). The anticipation is built into the physical structure and biochemical pathways that allows behavior change in a shorter time frame than is needed to attain the results that ensure survival. Here anticipation works to benefit the organism, since it acts *before* the outcome is certain, yet the behavior choice leads to a preferred task achievement. The behavior may change again *before* a negative effect is realized if the change in behavior was premature as cued by the niche, and thus not preferred at that time. Thus, the quick and malleable behavior choice is a trait of anticipation that leads to preferred task achievement, and it appears the organism *knew* the future before it actually occurred.

INFUSING ARCHETYPES

Although about 10 archetypes have been defined (Senge, 2006) or simplified for easier application (Wolstenholme, 2003, 2004), the two that most appropriate archetypes for use with simulating anticipation in robots are the Limits To Growth (LTG) and Shifting The Burden (or Goals) (STB) archetypes. The LTG archetype is appropriate since it approaches some asymptotic limit value based on some constraint in the system, as illustrated by the atomic behaviors described by Hayward and Boswell (2012).

The STB archetype has the attribute to change from a symptomatic solution to a more favorable fundamental solution with longer-term success. Unfortunately, the seemingly simpler short-term

symptomatic solution (SympSol) eventually leads to system failure. In contrast, the less obvious and likely more difficult to implement fundamental solution (FundSol) leads to a useful stable long-term solution. The key is to identify the archetype that is stuck in the repeated execution of a symptomatic solution headed for failure, and instead commit the resources needed to find and implement the process that carries out the fundamental solution. Realize the symptomatic solution is deceptive, because it actually seems to work in the short run, but does not in the long run. Ironically, the fundamental solution may not seem to have much of a positive effect in the short run, but increases the benefits the longer it is repeatedly applied. [This is ironic, since the choice for a symptomatic solution (creating failure) contradicts what is expected (success). It is *not* a paradox, since a paradox is a statement that contradicts itself, such as: "I always lie." See: http://english.stackexchange.com/questions/344071/the-difference-between-irony-and-paradox] Thus, adding the STB archetype to a basic system creates a desired solution, and moves from a simple reactive system to one that anticipates how to create an effective lasting solution. Such a system will have the traits of anticipation, acting *before* the outcome is certain, and yet reaching a result that appears to *know the future* before it happens.

ANTICIPATION ARCHITECTURE

A basic architecture was designed for the ANSIM (ANticipation SIMulation) model based on the causal loop and flow map diagrams that extract percepts from the niche environment, determined a suitable fitness level from a combination of factors, and used that along with threshold values to cue a behavior choice (Fig. 41). Specific coded processes represent the underlying routines, looping structure, and delays in which to place the notion of anticipation. The architecture can operate with No Anticipation (NO AN), or cued to operate with Anticipation On (AN ON).

Each behavior executes independently of the cue that makes it fire for operation, in the CHOICE BEHAVIOR (SCENARIO) module set, and choice of a behavior process excludes execution of another behavior process, meaning the behaviors are mutually exclusive. Explicitly, EXPLORE moves straight forward, AVOID turns counterclockwise (CCW), and SEEK turns CCW for a longer time period than AVOID and for a different cue. Each of the behaviors operates for a single time step. The overall process loops back for the next time step, and in the CONTEXT THRESHOLD determines the behavior choice for the next time step. Of course, the behavior choice is based on a suitable fitness value determined in the CONTEXT module based on the combination of values in the PERCEPT module. Percepts are measured in the NICHE environment to represent the current condition. Obviously, the current condition of the niche can change with each time step, thus resulting in a cue for a new behavior choice.

Anticipation in the CONTEXT module by includes methods for both the Limits To Growth (LTG) and Shifting The Burden (STB) archetypes. A WAIT behavior also was included in the BEHAVIOR CHOICE (SCENARIO) modules to allow for initialization at startup, and choice of a period of inactivity if unexpected conditions occur that do not match the three core behaviors (EXPLORE, AVOID, and SEEK).

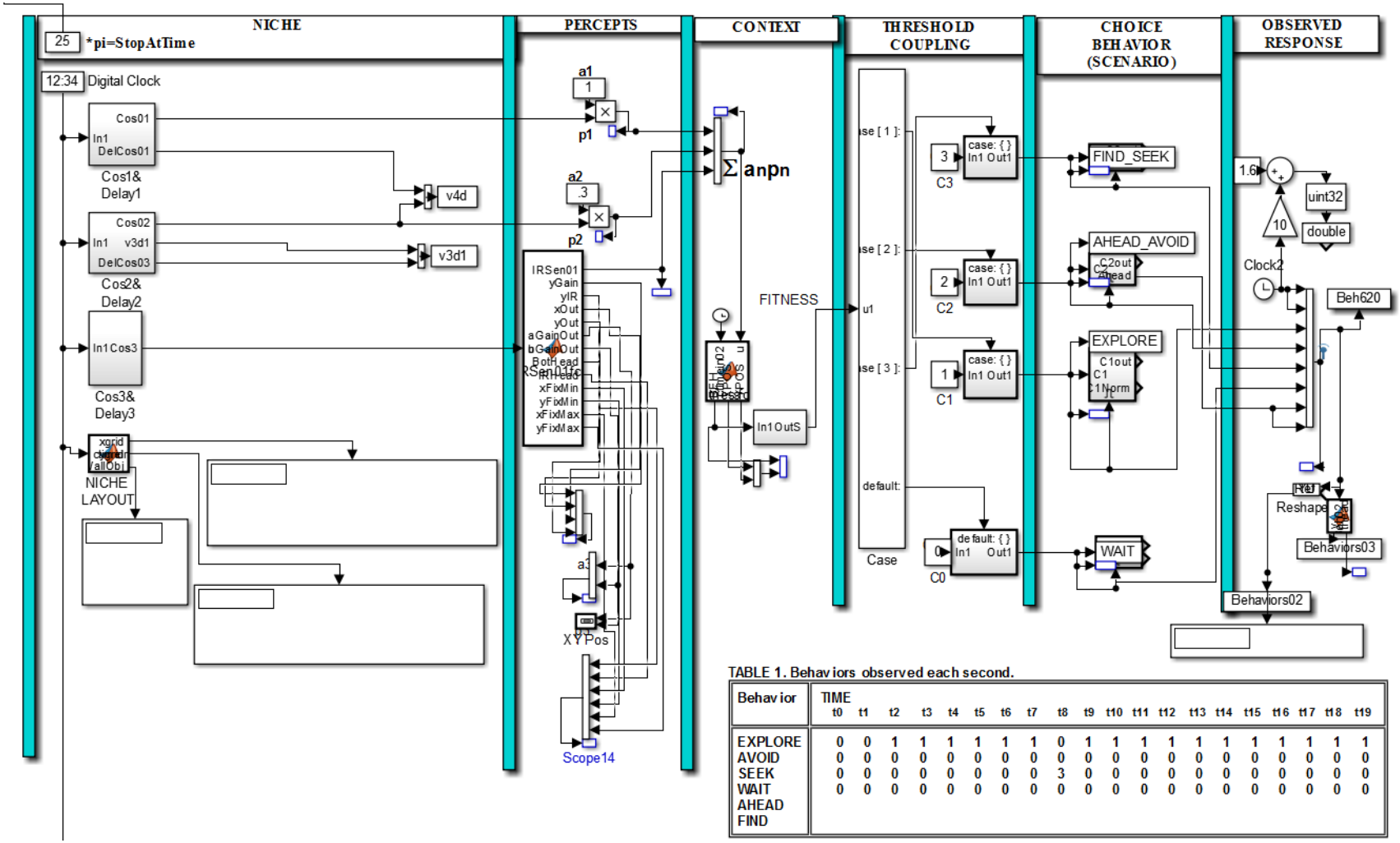


TABLE 1. Behaviors observed each second.

Behavior	TIME t0	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12	t13	t14	t15	t16	t17	t18	t19
EXPLORE	0	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
AVOID	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SEEK	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0
WAIT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AHEAD																				
FIND																				

Figure 41. Anticipation simulation model for TOURIST robot agent. Niche creates a 100 X 100 pixel arena within which the agent perceives the conditions (Percepts) and from that determines a Context fitness value that is used through Threshold Coupling to manifest a Choice Behavior that matches the current Niche conditions, thereby producing the Observed Response Behavior that leads to successful task achievement. Anticipation can be turned off or on as desired.

An AHEAD behavior was added in the CONTEXT module to reflect the system dynamics of the LTG archetype. AHEAD measures a percept of up to a maximum distance, a distance that is greater than needed for a turn to miss an object or wall, and scales the timing of the subsequent AVOID behavior relative to a minimum safe turn distance (28 cm). The scaled timing of AVOID may execute for several brief time steps within the model looping structure. In this way, a smaller turn is made than would occur for the case if no anticipation is on (no AN ON). This allows for the EXPLORE behavior to be cued again in a shorter time span than for no anticipation on. The overall result is a series of smaller AVOID turn adjustments with integrated EXPLORE forward behavior. Thus, the TOURIST responds to an object that is farther away in anticipation of a future encounter. The result is still to perform the task to travel along an extended wall, response to and miss objects, and make abrupt changes when a new wall is encountered based in a different orientation (north and south versus east and west).

Anticipation also was included by adding the FIND behavior that also was placed in the CONTEXT module using the system dynamics of the STB archetype. FIND also uses the percept of a maximum distance (50 cm) that is greater than needed for a safe turn (minimum 28 cm) to miss an object or wall, and uses that detection to halt (interrupt) the SEEK behavior when it is actively executing. With anticipation on (AN ON), the interrupt of SEEK allows cue of the EXPLORE behavior to initiate a move towards the perceived new object, so the SEEK behavior no longer is set to a mandatory turn of a total of 270 degrees CCW, but may halt the turn at any point in rotation that an object is detected. The halt of SEEK and cue of EXPLORE may occur in any time step within the normal period needed for the 270 degree turn. Thus, the time for a spin turn in SEEK may be reduced if an object is close enough for the TOURIST to FIND it within

the maximum set distance. Once EXPLORE has been cued, in subsequent time steps the AHEAD and AVOID behaviors operate to make the small adjustments to again follow a wall or miss an object. The result of FIND is to locate the nearest object or wall, and to begin to undertake a new heading toward that object or wall. Thus, the maximum energy as was used by SEEK is reduced if another option is found. However, the wall that was previously being followed, though not detected for a set time (to cue SEEK), would be abandoned for the newly found wall. However, if no object is in the maximum distance range during the spin of a SEEK behavior, the TOURIST would again encounter and follow the previous wall.

VERIFIED SYSTEM DYNAMICS SIMULATION

Anticipation was applied in simulation for a systems architecture in a simple square arena that could contain no objects or multiple objects. Coded behaviors were matched to the arena boundaries and the objects inside, with intended observed criteria to be to follow the arena boundary walls or move near yet not contact the objects for successful task achievement. The robot agent was located at various initial (x,y) locations with chosen heading and forward motion, and the resulting path was tracked and plotted for the travel within the arena.

Time Step and Sensing

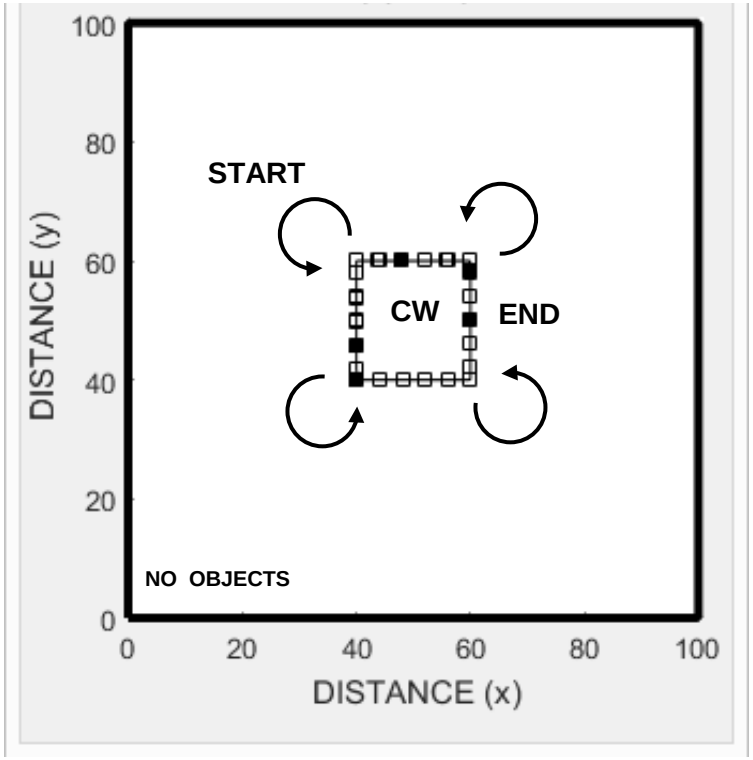
Repeated simple relatively short time tests were run to verify the simulated agent performed as would be expected in a real-world arena. Initial testes were successful at creating expected behaviors and task achievement of wall following and avoidance of objects. However, on rare occasions the agent seemed to escape from the bounded arena., and the suspect behavior could be recreated by starting with known settings that led to the escapes. Further examination of the

coded procedures revealed the two undesired traits: first, the time step resolution was not small enough at 0.1 sec steps to capture certain timed sequences correctly, and second the model of the infrared (IR) sensor perception was not refined enough to truly capture a distance measurement that would be realized from a real-world IR sensor. The time step was reduced to 0.01 sec at the expense of increased simulation time, yet the tracking of behavior and motion was considerably improved. For distance sensing, the testing of increments in a straight line for the sensor was refined to ensure all cells (or pixels) were sampled to find a wall or object. After these simple changes, the agent always remained in the arena, and had reactions to the walls and objects as desired and expected. Note that these problems were traits of the simulation, and not of the behavior responses chosen, meaning that proper description of the virtual simulation environment was the source of errors rather than simulation of agent response to the environment. A real-world agent with correctly operational sensors would be able to perform the assigned behaviors.

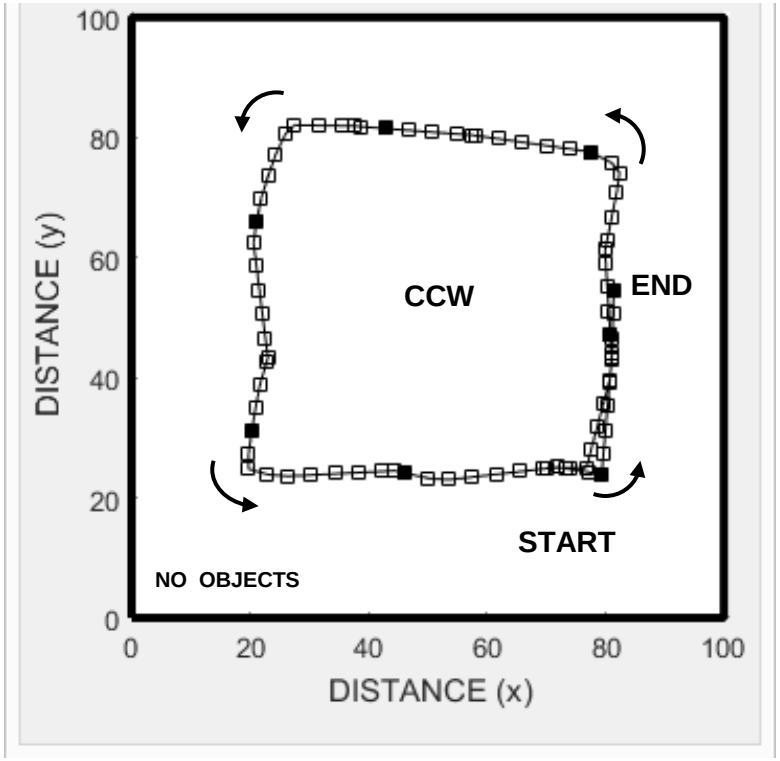
SEEK behavior

Simulation allows for precise geometric timing responses for behavior. Based on the original strategy from Connell (1990), the simulated SEEK behavior should rotate 270 degrees CCW, and that effectively looks like a 90 degree CW rotation in a path drawing. Placement of the agent near the middle of the arena with a heading away from a wall (and no objects in the arena) and with a short enough time delay before a cue of SEEK should create an unending square path near the middle of the arena. The path created in simulation was found to agree with this expectation, and provided evidence the simulation model is verified to perform correctly as one would

observe in a real-world arena (Fig. 42, left). Greater detail of the dynamics of the travel show the SEEK cue occurs repeatedly over time (Fig. 43, right).

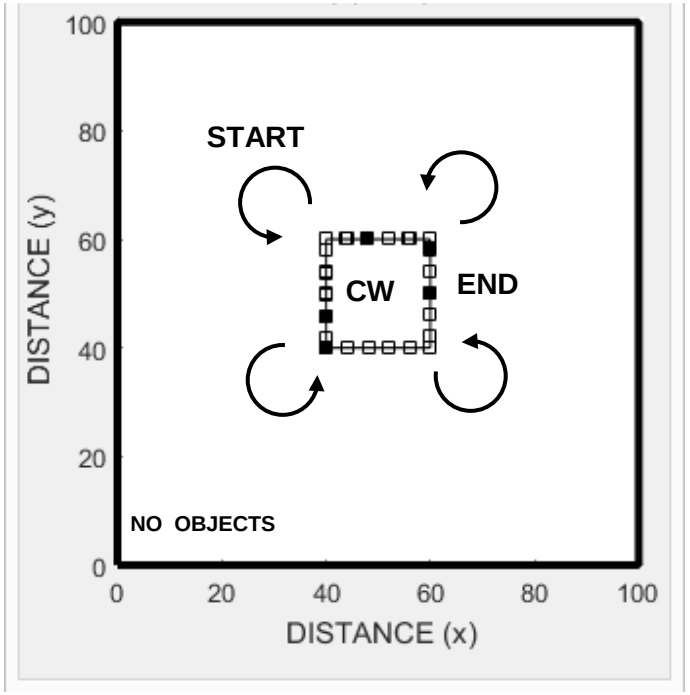


NO ANTICIPATION. 28 cm



NO ANTICIPATION. 28 cm

Figure 42. AN path simulated for 78 s (25*pi). No Anticipation: 28 cm; Initial: xpos=40, ypos=60 (left); No Anticipation on: 28 cm, Initial: xpos=70, ypos=25 (right);. When Minimum= Maximum distance= 28 cm, there is effectively No Anticipation; When starting in the upper left, yet near the middle, 270 deg CCW turns (90 deg CW) for SEEK make a perfect square path, yet the robot is trapped in space indefinitely (left). When starting in the lower right near the wall, a roughly square path follows near the wall arena boarder, again indefinitely.
SEEK not interrupted (every 10th pt. darker) Path: start heading= 0 deg.; Initial: xpos=varies, ypos=varies, tAVOID=0.125s, tWaitSEEK=5s, NicheLayout=100X100cm, IROffset= 30 deg, speed= 15 cm/s; IRDistMin= 28 cm; IRDistMax= 28 cm.

a. 

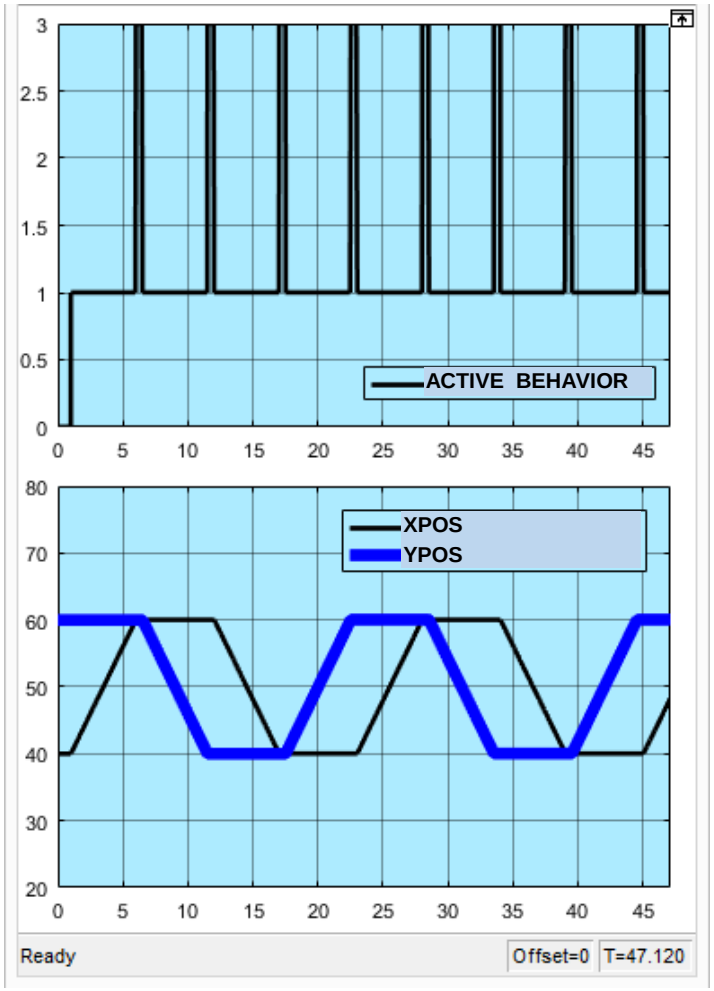
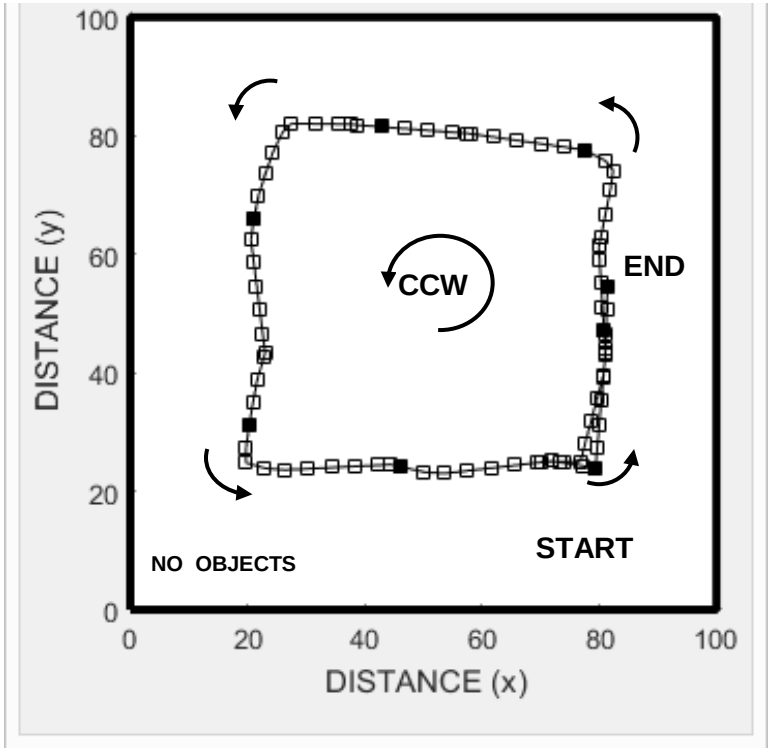
b 

Figure 43. TOURIST No AN: path simulated for 47 s (15*pi). a. Arena bounded with no internal objects b. Behaviors over time and locations in x-y plane. The TOURIST turns CCW at each corner as SEEK wait time cues, so 270 degree CCW rotation actually turns 90 CW each time, and near middle of the arena no walls are ever found:'stranded' in space. SEEK not interrupted (10th pt. darker) Start heading= 0 deg.; Initial: xpos=40, ypos=60, tAVOID=0.125s, tWaitSEEK=5s, NicheLayout=100X100cm, IROffset= 30 deg, speed= 15 cm/s; IRDistMin= 28 cm; IRDistMax= 60 cm.

a. 

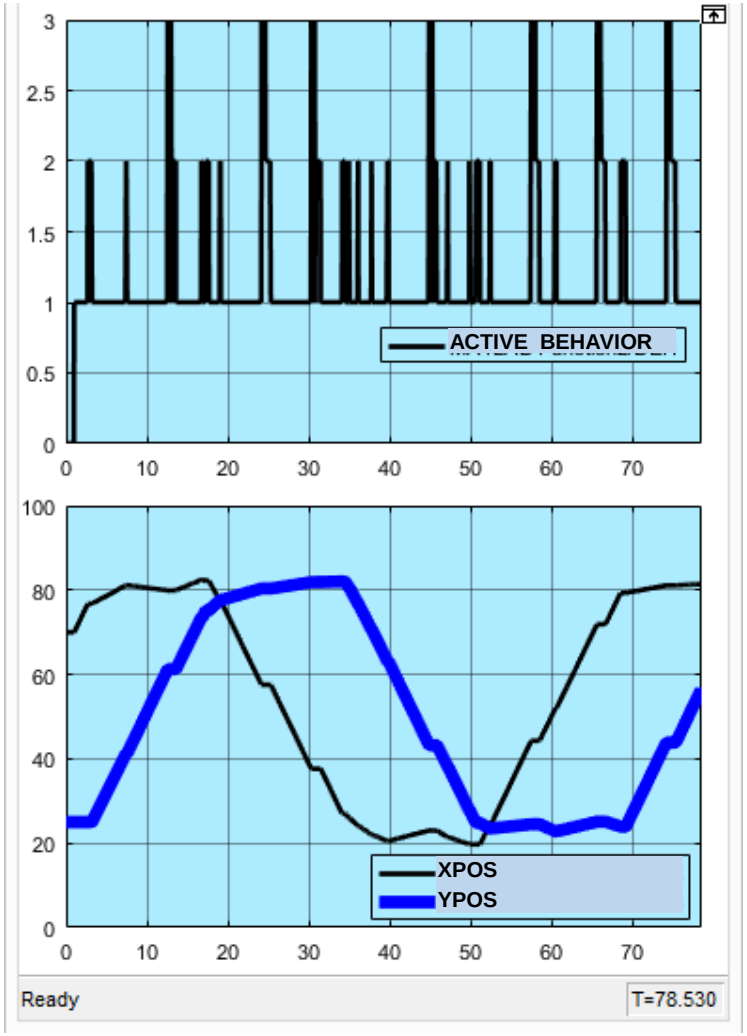
b 

Figure 44. TOURIST No AN: path simulated for 78 s (25*pi). a. Arena bounded with no internal objects and shown path; b. Behaviors over time and locations in x-y plane. The TOURIST is near the wall boarder, so AVOID & SEEK work together to follow the wall. SEEK not interrupted (every 10th pt. darker) Start heading= 0 deg.; Initial: xpos=70, ypos=25, tAVOID=0.125s, tWaitSEEK=5s, NicheLayout=100X100cm, IROffset= 30 deg, speed= 15 cm/s; IRDistMin= 28 cm; IRDistMax= 60 cm.

AVOID and SEEK Behavior Mixed

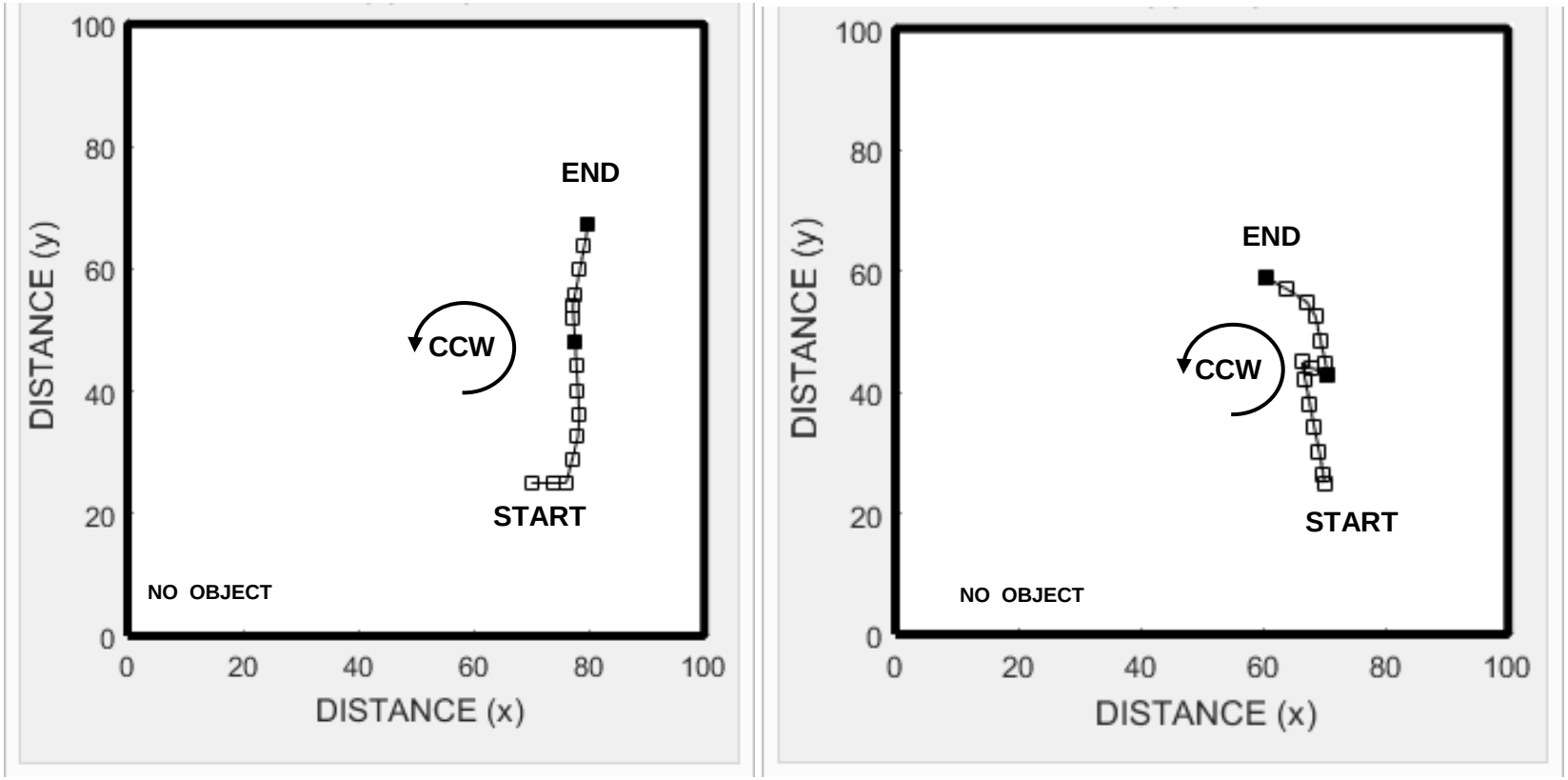
A simple application of the AVOID behavior results from initial placement of the agent near a wall (in an arena with no objects) and allowing for continuous operation near the arena boundary wall. Proper AVOID behavior maintains a minimal distance from the wall, while SEEK invokes on occasion to move back closer to the wall. Again, the path created in simulation was found to agree with this expectation, and provided evidence the simulation model is verified to perform correctly as one would observe in a real-world arena (Fig. 42, right). Additional detail of the path traveled shows both AVOID and SEEK behaviors work together to follow the wall, and the path nearly overlaps as a full cycle is traveled around the arena (Fig. 44, left).

Anticipation Simulation: AVOID and SEEK Behavior Mixed

A relatively brief time run of the simulation with No Anticipation or with Anticipation On shows the nature of the addition of anticipation (Figs. 45, 46, & 47). With no anticipation (NO AN), the simply reactive behavior merely travels a path closely along the wall. In contrast, with anticipation on (AN ON), the TOURIST reacts to a wall in the distance that will be encountered in the future, and makes smaller adjustments to trend away from the wall before any direct encounter is realized. Several comparisons were made to show the results with or without anticipation (Figs. 50 to 62)

Brief runs with anticipation off or on was used to illustrate the effect of increasing the maximum distance used for calculation of anticipation in the AHEAD and FIND behaviors (Figs. 48 & 49). Again, no anticipation has the TOURIST merely follow closely along the wall. With anticipation on, increasing the maximum distance moderately has marginally increased reaction effects.

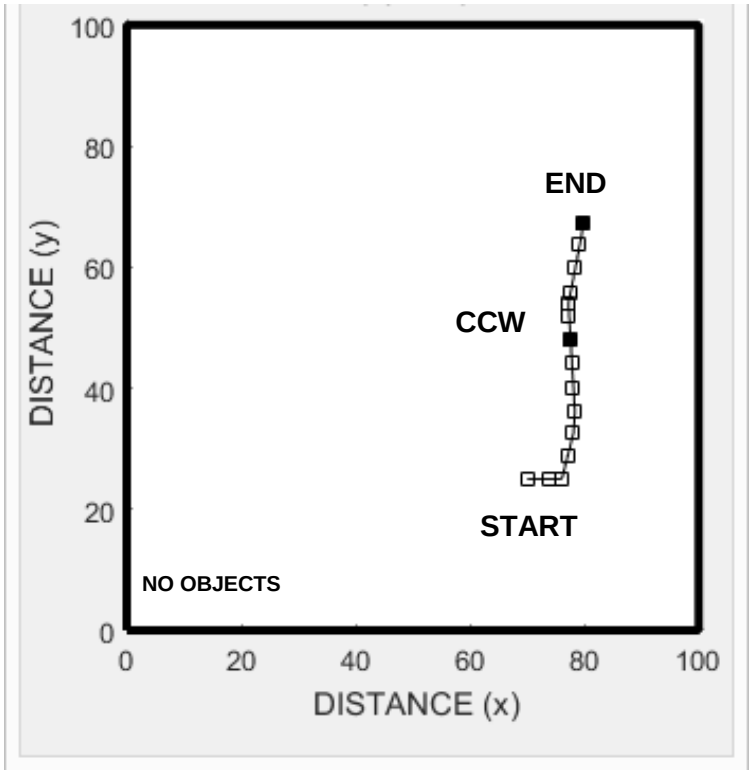
Perceived distant objects cause behavior to adjust earlier to avoid and follow along a wall. As this extends to farther maximum distances (greater ranges), eventually the effect is premature, and over-anticipation of future object or wall encounters actually prohibits the TOURIST from effectively navigating the arena. As with many favorable items, it is possible to have too much of a good thing. So it is best to limit the maximum distance (to perhaps 50 cm), and thus ultimate range considered, so as not to react so early that a reasonable result does not occur. This agrees with the contention of Simon (1996), that items too distant in space or time should not be considered in design, or in practical application.



NO ANTICIPATION. 28 cm **ANTICIPATION ON. 28 & 50 cm**

Figure 45. AN path simulated for 15 s (5*pi). No Anticipation: 28 cm; (left); Anticipation On: 28 cm (right), Both initial: xpos=70, ypos=25 (right);. When Minimum= Max = 28 cm, there is effectively No Anticipation; Left: No Anticipation: The TOURIST is near the wall boarder, so AVOID & SEEK work to follow the wall.
Right: Anticipation moves away from the expected wall, and after the delay, SEEK cues at 6.76 s, yet before 270 deg is turned, at 7.08 s a wall is found by FIND, and the TOURIST moves towards it, combining with AHEAD & AVOID to again move along the wall. No other SEEK & FIND behavior is cued over the interval. Path: start heading= 0 deg.; Initial: xpos=varies, ypos=varies, tAVOID=0.125s, tWaitSEEK=5s, NicheLayout=100X100cm, IROffset= 30 deg, speed= 15 cm/s; IRDistMin= 28 cm; IRDistMax= 28 & 50 cm.

a.



b

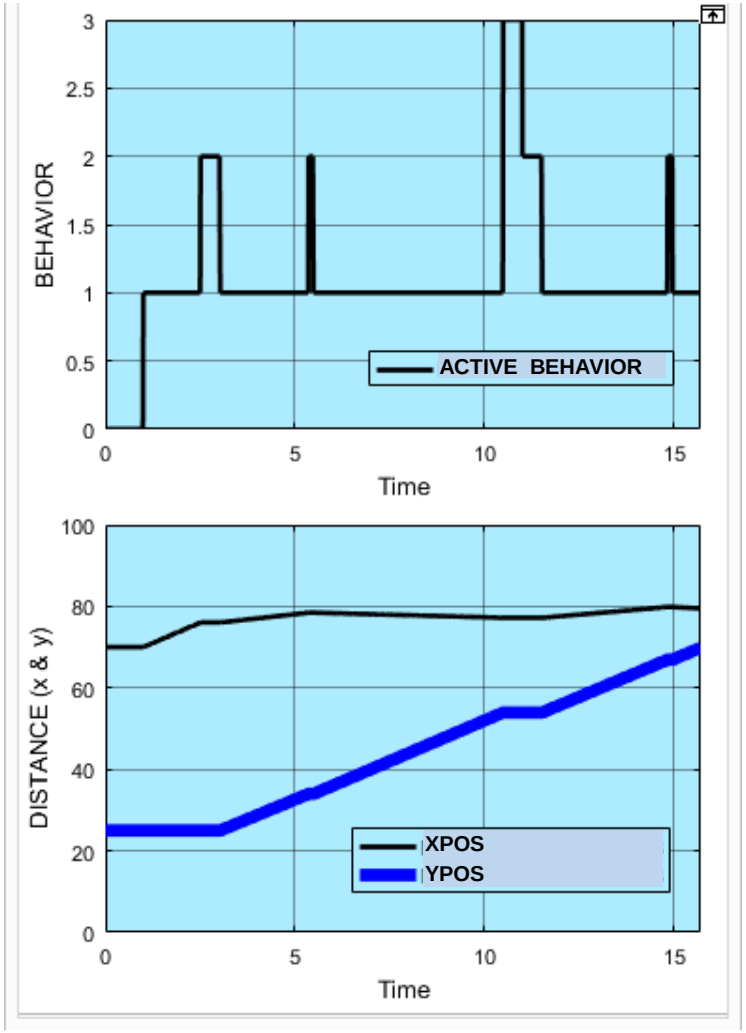
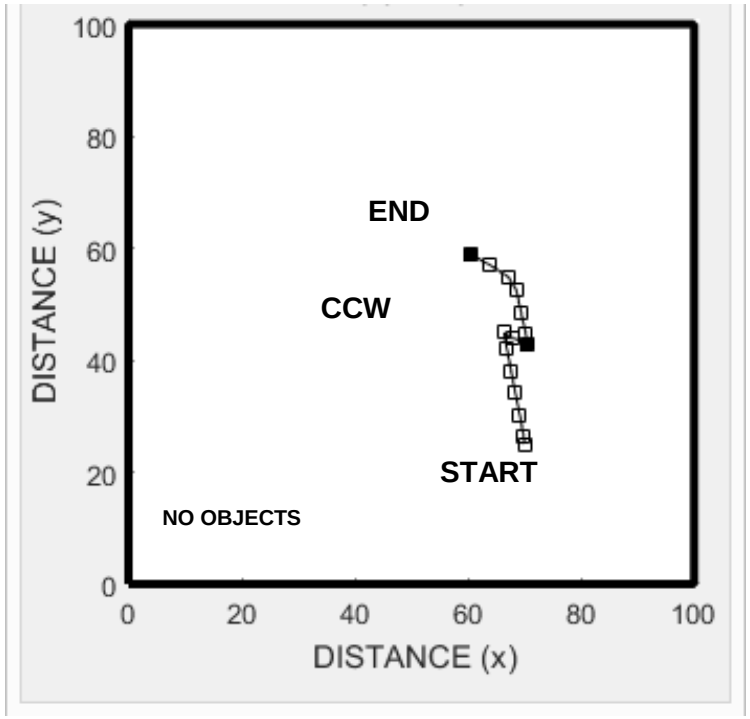
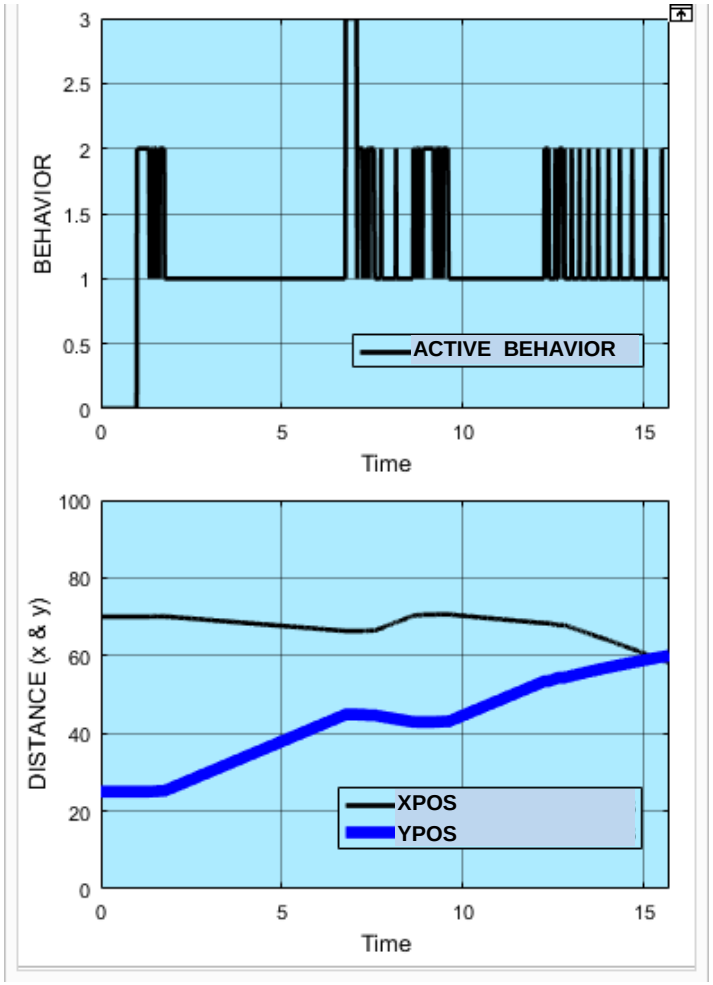


Figure 46. TOURIST No AN: path simulated for 15 s (5*pi). a. Arena bounded with no internal objects and shown path; b. Behaviors over time and locations in x-y plane. The TOURIST is near the wall boarder, so AVOID & SEEK work to follow the wall. SEEK not interrupted (every 10th pt. darker) Path: start heading= 0 deg.; Initial: xpos=70, ypos=25, tAVOID=0.125s, tWaitSEEK=5s, NicheLayout=100X100cm, IROffset= 30 deg, speed= 15 cm/s; IRDistMin= 28 cm; IRDistMax= 28 cm.



a.



b

Figure 47. TOURIST Anticipation ON: path simulated for 15 s (5*pi). a. Arena bounded with no internal objects and path shown; b. Behaviors over time and locations in x-y plane. Anticipation moves away from the expected wall, and after the delay, SEEK cues at 6.76 s, yet before 270 deg is turned, at 7.08 s a wall is found and TOURIST moves towards it, combining with AHEAD & AVOID to again move along the wall. No other SEEK & FIND behavior is cued over the interval. Path: start heading= 0 deg.; Initial: xpos=70, ypos=25, tAVOID=0.125s, tWaitSEEK=5s, IROffset= 30 deg, speed= 15 cm/s; IRDistMin= 25 cm; IRDistMax= 50 cm.

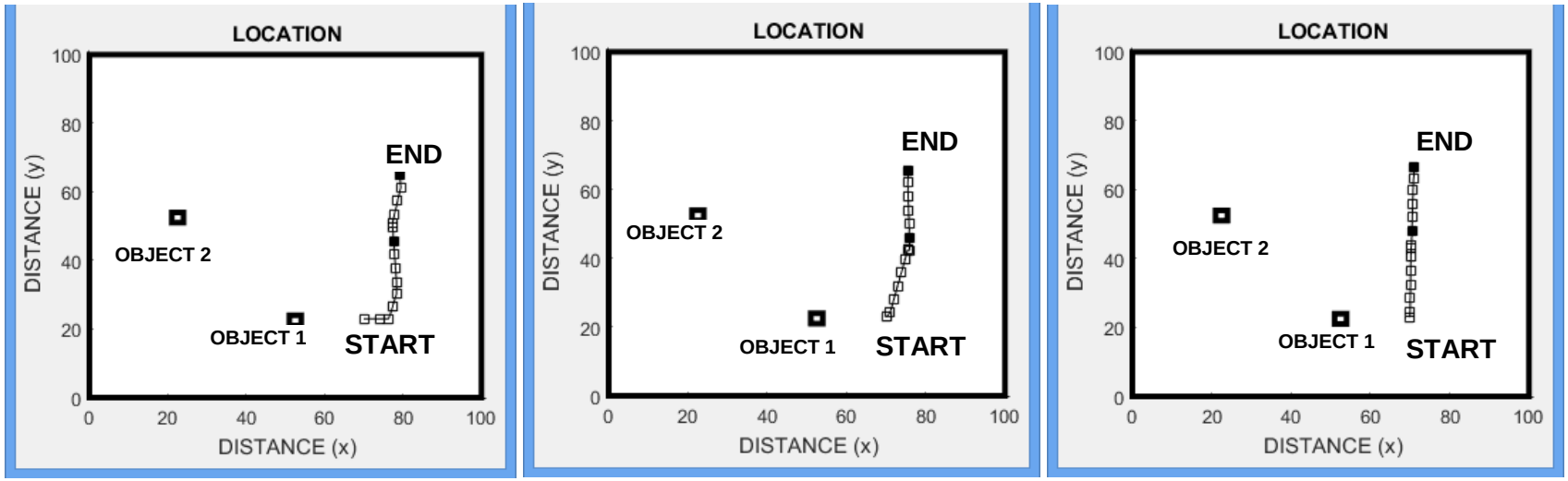
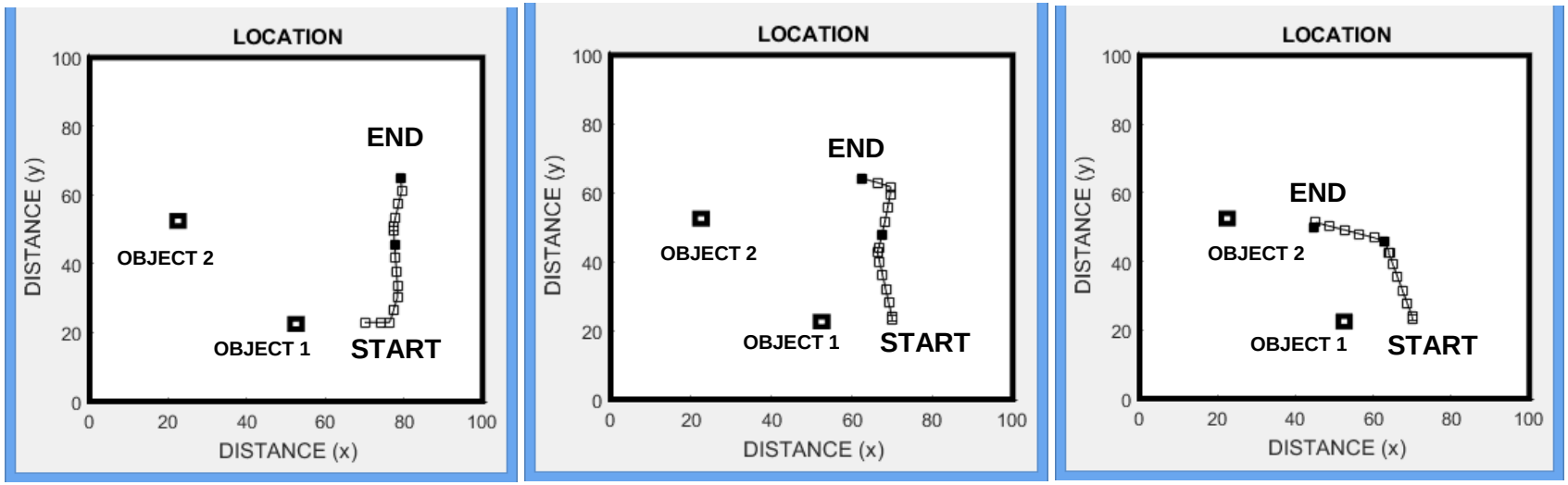
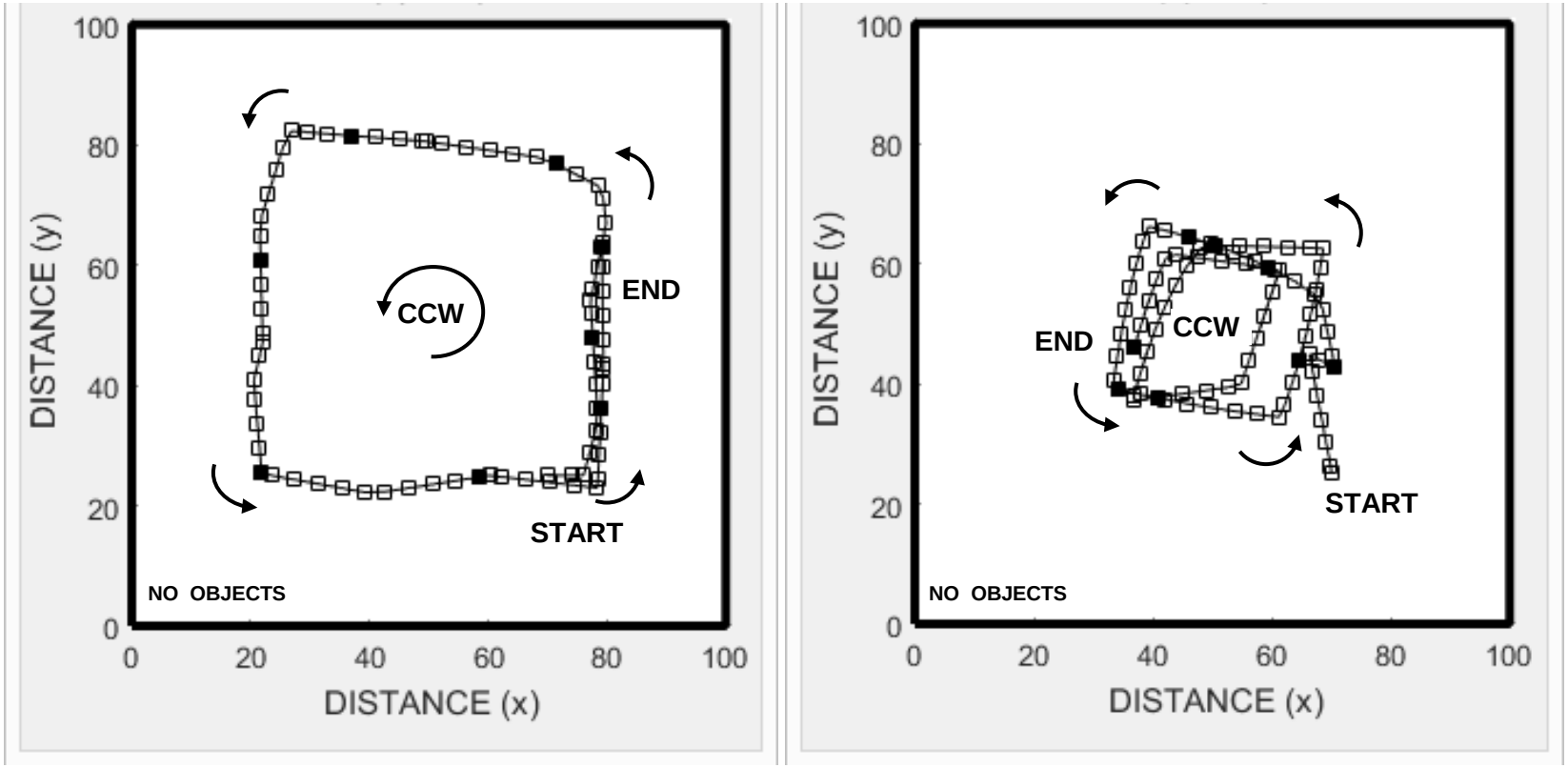


Figure 48. AN path simulated for 15 s (5*pi). No anticipation (left); Anticipation on: 28 & 35 cm (middle); Anticipation on: 28 & 40 cm. Minimum distance is 28 cm for all; Looks AHEAD further to distance of 35 or 40 cm. Perceived distant objects cause behavior to adjust earlier to avoid and follow along a wall.
SEEK not interrupted (every 10th pt. darker) Path: start heading= 0 deg.; Initial: xpos=70, ypos=23, tAVOID=0.125s, tWaitSEEK=5s, NicheLayout=100X100cm, IROffset= 30 deg, speed= 15 cm/s; IRDistMin= 28 cm; IRDistMax= varies cm.



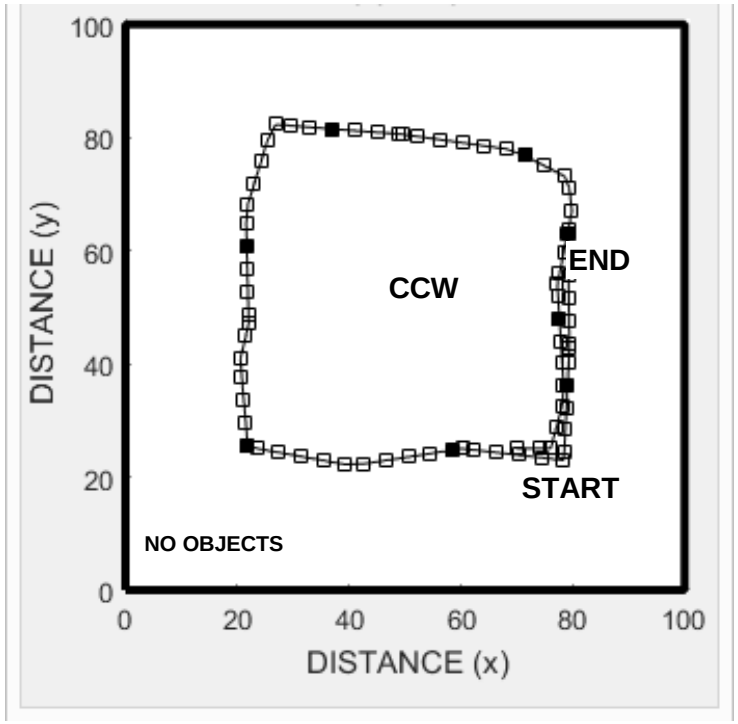
NO ANTICIPATION. 28cm **ANTICIPATION ON. 28&50cm** **ANTICIPATION ON. 28&60cm**

Figure 49. AN path simulated for 15 s (5*pi). No anticipation (left); Anticipation on: 28 & 50 cm (middle); Anticipation on: 28 & 60 cm. Minimum distance is 28 cm for all; Looks AHEAD further to distance of 50 or 60 cm. Cases show proper Anticipation (middle), and OVER-Anticipation (right) that is reacting too early and in extreme to objects perceived in the distance.
SEEK not interrupted (every 10th pt. darker) Path: start heading= 0 deg.; Initial: xpos=70, ypos=23, tAVOID=0.125s, tWaitSEEK=5s, NicheLayout=100X100cm, IROffset= 30 deg, speed= 15 cm/s; IRDistMin= 28 cm; IRDistMax= varies cm.

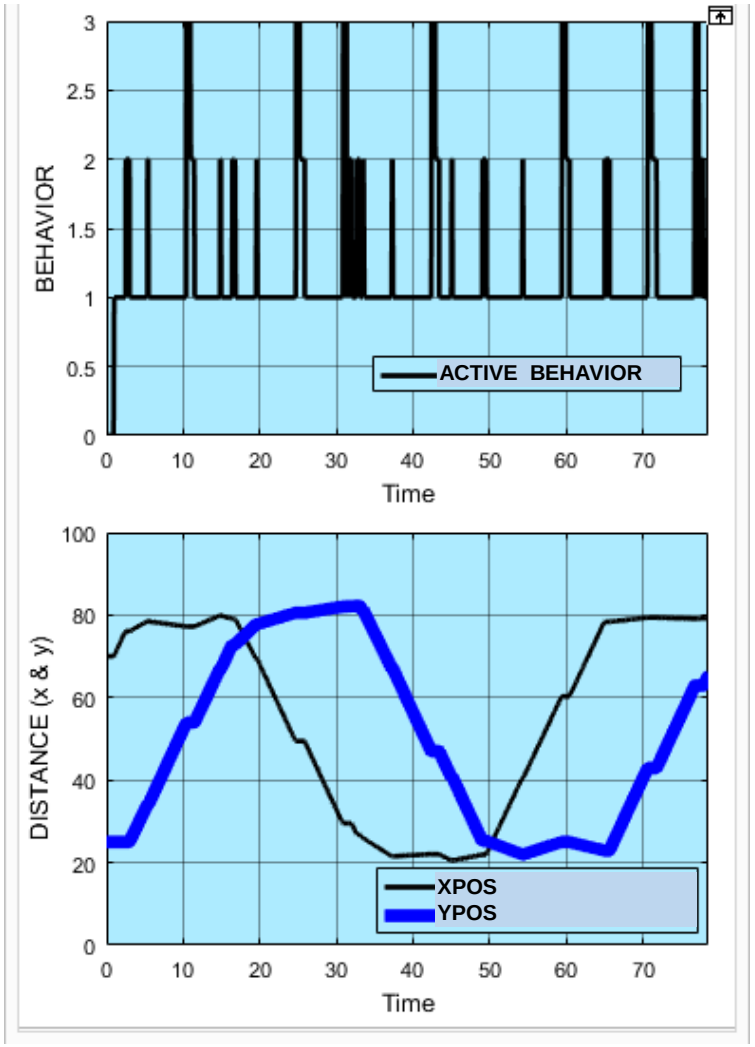


NO ANTICIPATION. 28 cm **ANTICIPATION ON. 28 & 50 cm**

Figure 50. AN path simulated for 78 s (25*pi). No Anticipation: 28 cm; Anticipation On: 28 cm (right); Both initial: xpos=70, ypos=25;. When Minimum= Maximum distance= 28 cm, there is effectively No Anticipation;
Left: No Anticipation: Combined AHEAD & AVOID move along the wall; after the delay, SEEK cues at 10.5s, & 6 later times.
Right: Anticipation moves away from the expected wall, and after the delay, SEEK cues at 6.76 s, yet before 270 deg is turned, at 7.08 s a wall is found and TOURIST moves towards it, combining with AHEAD & AVOID to again move along the wall. Two more SEEK & FIND behaviors are cued over the interval shown. Dynamic changes in movement are more gradual and preemptive with Anticipation, or abrupt if Anticipation shows that is needed, as in corners and a new wall encounter.
Path: start heading= 0 deg.; Initial: xpos=varies, ypos=varies, tAVOID=0.125s, tWaitSEEK=5s, IROffset= 30 deg, speed= 15 cm/s; IRDistMin= 28 cm; IRDistMax= 28 & 50 cm.



a.



b

Figure 51. TOURIST NO AN: path simulated for 78 s (25*pi). a. Arena bounded with no internal objects and path shown; b. Behaviors over time and locations in x-y plane. Combined AHEAD & AVOID moves along the expected wall, and after the delay, SEEK cues at 10.5 s, and at 6 later times. Path: start heading= 0 deg.; Initial: xpos=70, ypos=25, tAVOID=0.125s, tWaitSEEK=5s, IROffset= 30 deg, speed= 15 cm/s; IRDistMin= 28 cm; IRDistMax= 28 cm.

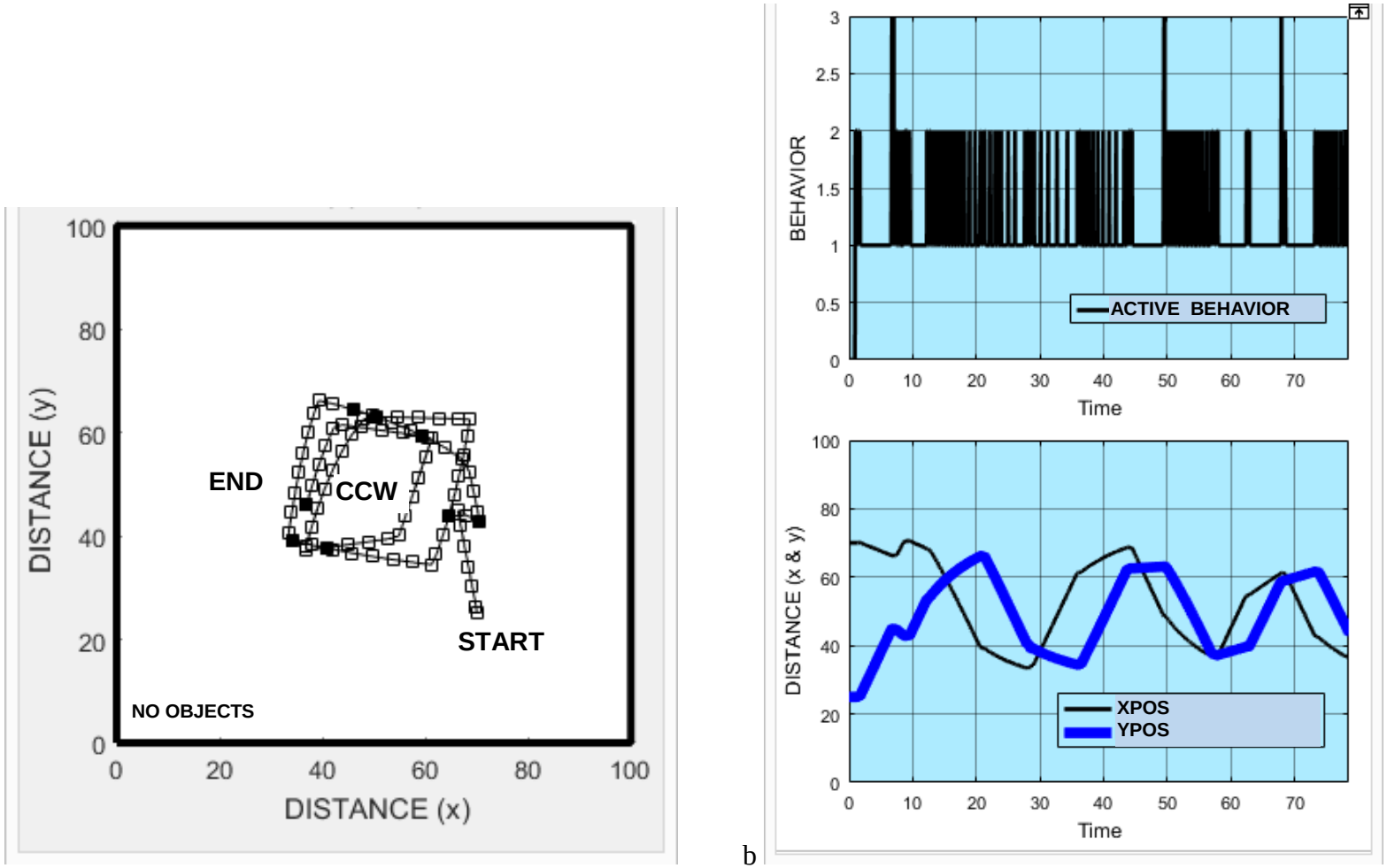
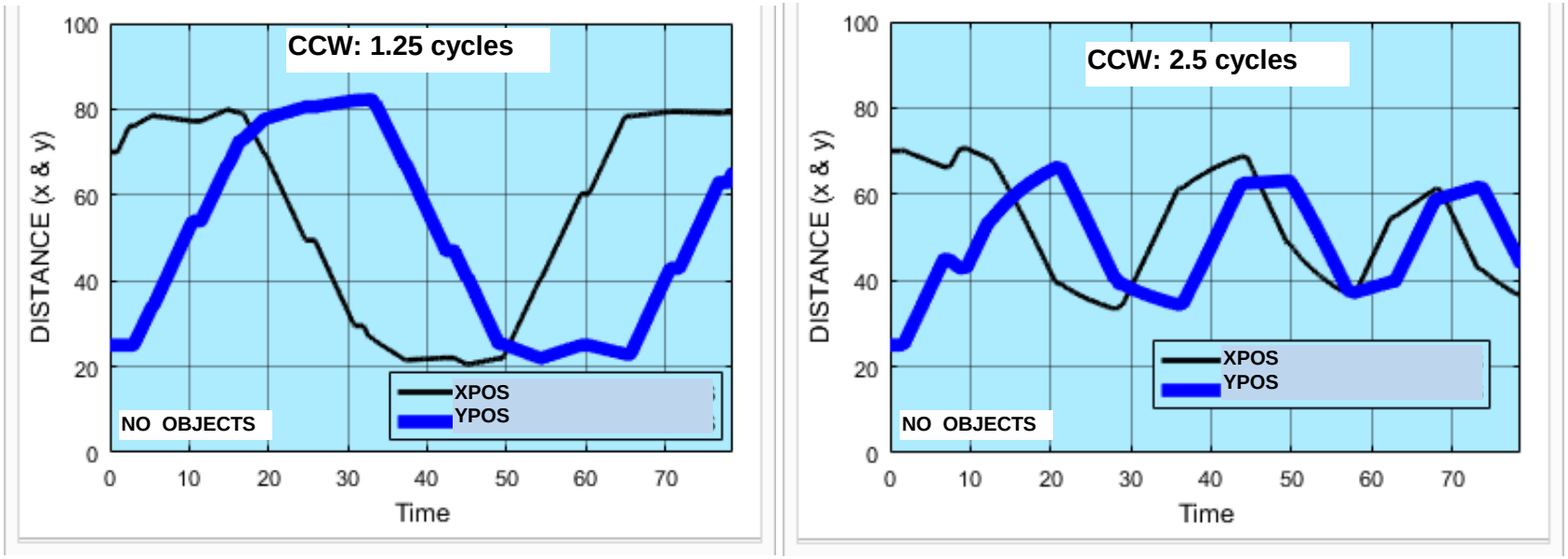


Figure 52. TOURIST AN ON: path simulated for 78 s (25*pi). a. Arena bounded with no internal objects and path shown; b. Behaviors over time and locations in x-y plane. Anticipation moves away from the expected wall, and after the delay, SEEK cues at 6.76 s, yet before 270 deg is turned, at 7.08 s a wall is found and TOURIST moves towards it, combining with AHEAD & AVOID to again move along the wall. Two more SEEK & FIND behaviors are cued over the interval shown. Path: start heading= 0 deg.; Initial: xpos=70, ypos=25, tAVOID=0.125s, tWaitSEEK=5s, IROffset= 30 deg, speed= 15 cm/s; IRDistMin= 28 cm; IRDistMax= 50 cm.



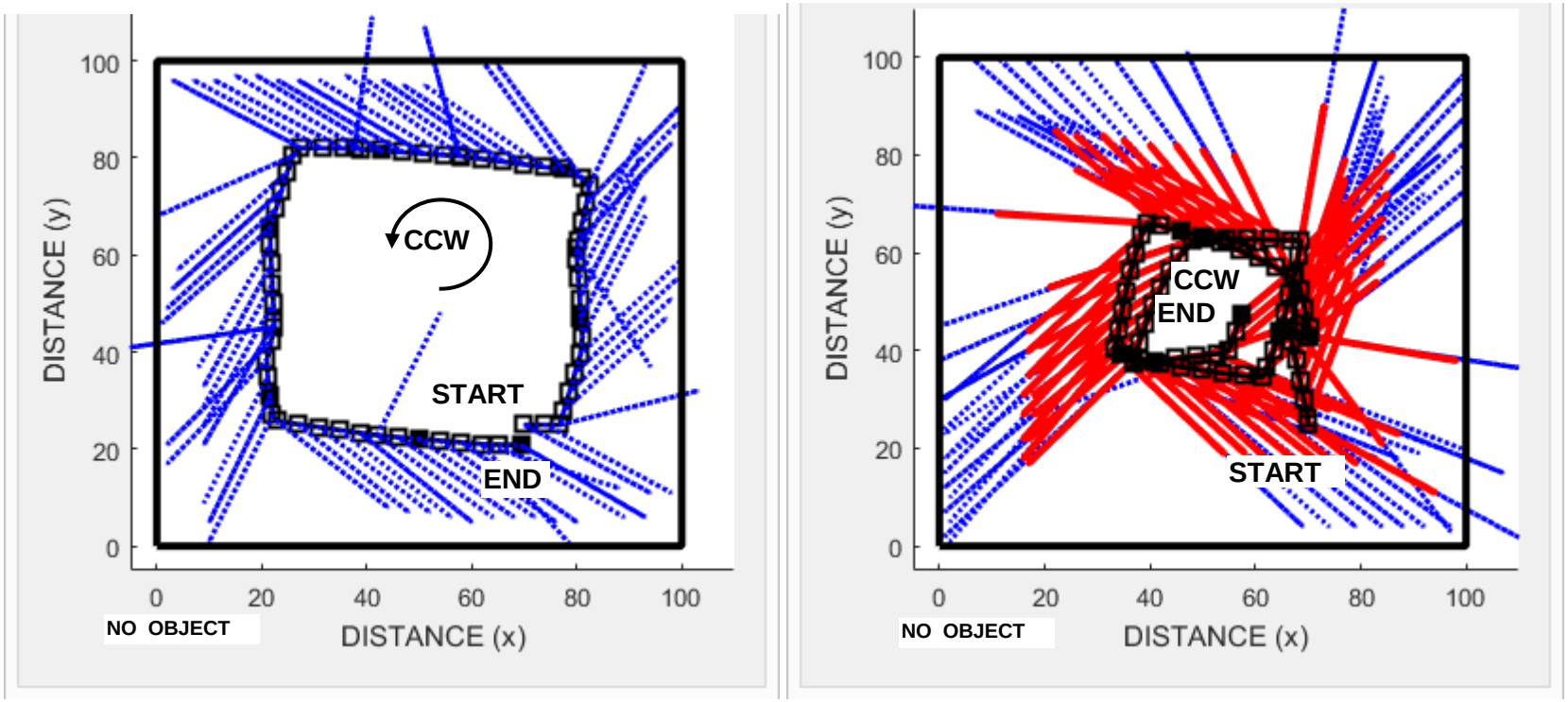
NO ANTICIPATION. 28 cm **ANTICIPATION ON. 28 & 50 cm**

Figure 53. AN path simulated for 78 s (25*pi). No Anticipation: 28 cm (left); Anticipation On: 28 cm (right); Both initial: xpos=70, ypos=25;. When Minimum= Maximum distance= 28 cm, there is effectively No Anticipation; Twice as many cycles occur with Anticipation On (Right: 2.5 cycles) vs. No Anticipation (Left: 1.25 cycles), thus permitting more rapid and thorough coverage over time for the arena. Both travel the same forward speed: 15 cm/s.

Left: No Anticipation: Combined AHEAD & AVOID move along the wall; after the delay, SEEK cues at 10.5 s, and at 6 later times.

Right: Anticipation moves away from the expected wall, and after the delay, SEEK cues at 6.76 s, yet before 270 deg is turned, at 7.08 s a wall is found and TOURIST moves towards it, combining with AHEAD & AVOID to again move along the wall. Two more SEEK & FIND behaviors are cued over the interval shown.

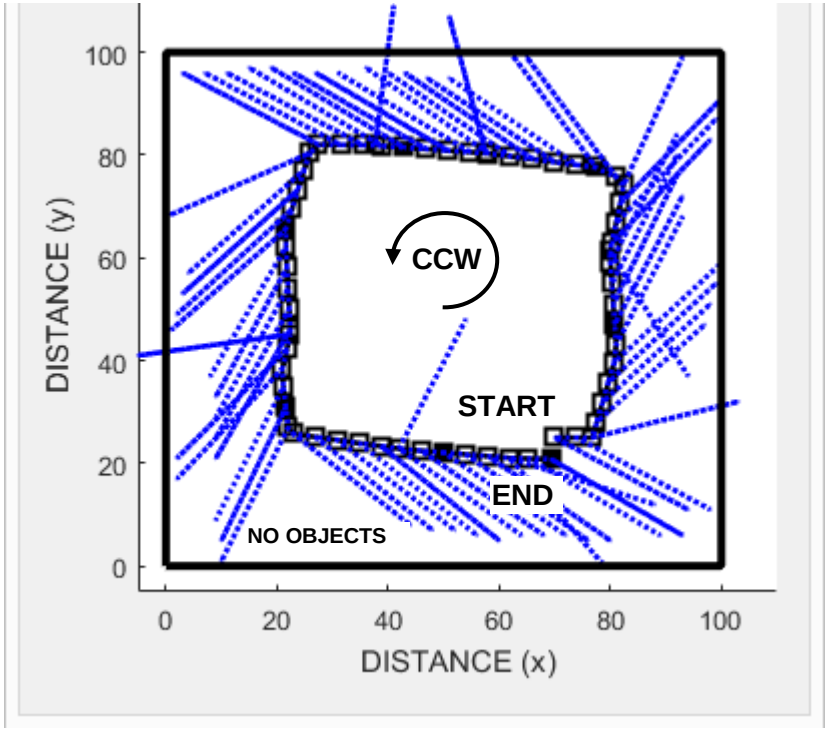
Path: start heading= 0 deg.; Initial: xpos= 70, ypos=25, tAVOID=0.125s, tWaitSEEK=5s, IROffset= 30 deg, speed= 15 cm/s; IRDistMin= 28 cm; IRDistMax= 28 & 50 cm.



NO ANTICIPATION. 28 cm **ANTICIPATION ON. 28 & 50 cm**

Figure 54. AN path simulated for 66 s (21*pi). No Anticipation: 28 cm; (left); No Anticipation on: 28 cm, Both Initial: xpos=70, ypos=25 (right);. Time shown of 66 s is for about one cycle around the area with No AN, but results in about 2 cycles with AN On. Dotted lines show path of IR sensor percept beam at each second from the robot location to the IR endpoint 50 cm away, darker line only 28 cm. Random appearing directions occur during the SEEK routine spin that may point in almost any direction, but is only used when AN in On. Dynamic changes in movement are more gradual and preemptive with Anticipation ON, or abrupt if Anticipation shows that is needed, as in corners and a new wall encounter. Two cycles are made around the arena with AN On, versus only one cycle with No AN, lending for more area covered and more dynamic discovery.
Path: start heading= 0 deg.; Initial: xpos=varies, ypos=varies, tAVOID=0.125s, tWaitSEEK=5s, NicheLayout=100X100cm, IROffset= 30 deg, speed= 15 cm/s; IRDistMin= 28 cm; IRDistMax= 28 cm.

a.



b

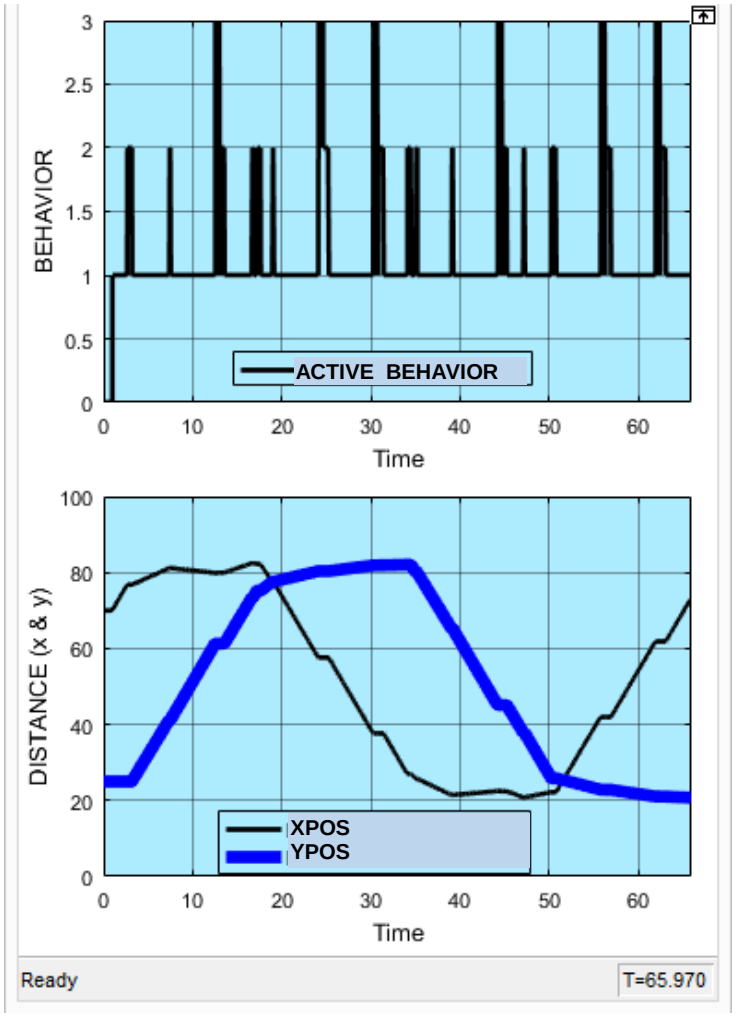
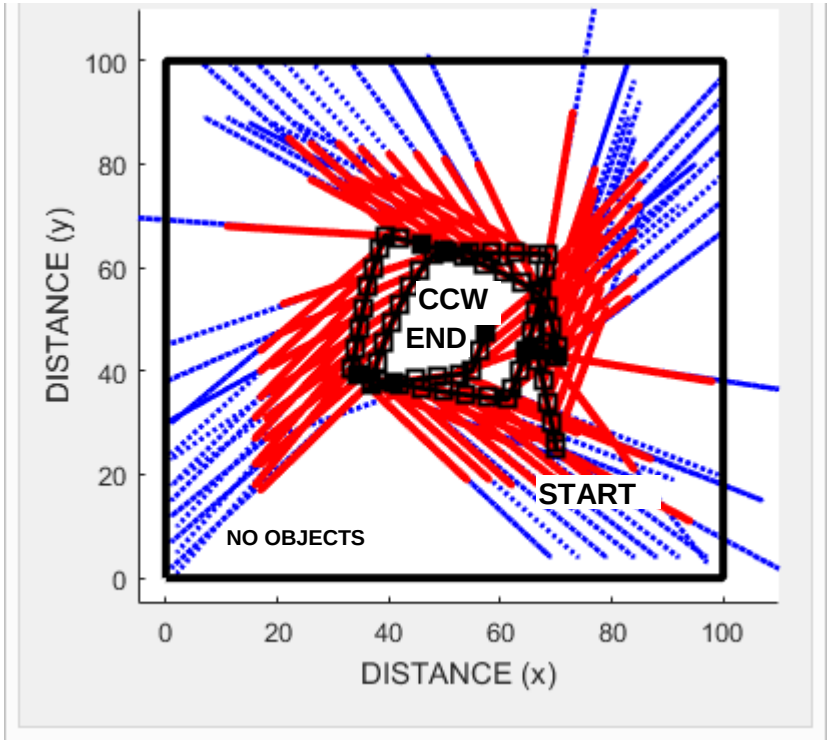


Figure 55. TOURIST No AN: path simulated for 66 s (21*pi). a. Arena bounded with no internal objects and path shown; b. Behaviors over time and locations in x-y plane. Combined AHEAD & AVOID moves along the expected wall, and after the delay, SEEK cues at about 13 s, and at 5 later times. Path is one full cycle around the arena. SEEK interrupted (every 10th pt. darker) Path: start heading= 0 deg.; Initial: xpos=70, ypos=25, tAVOID=0.125s, tWaitSEEK=5s.

a. 

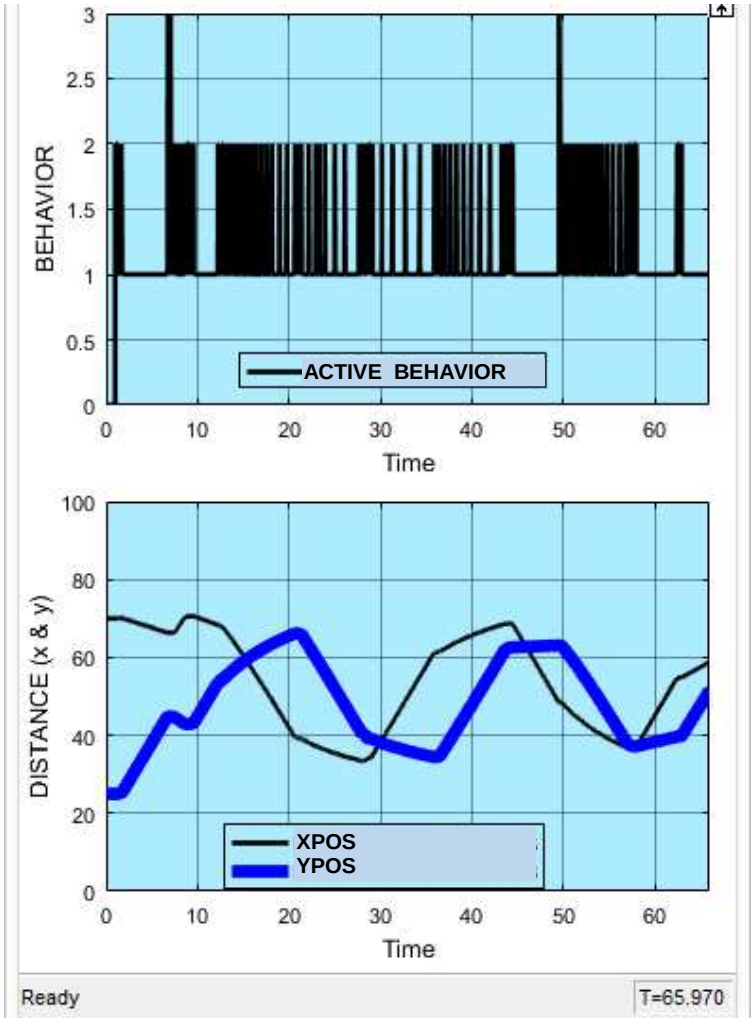
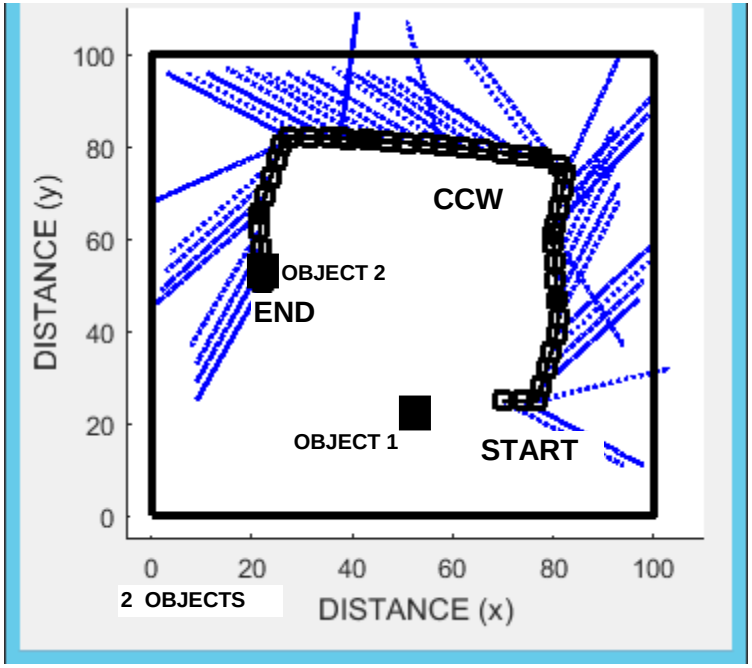
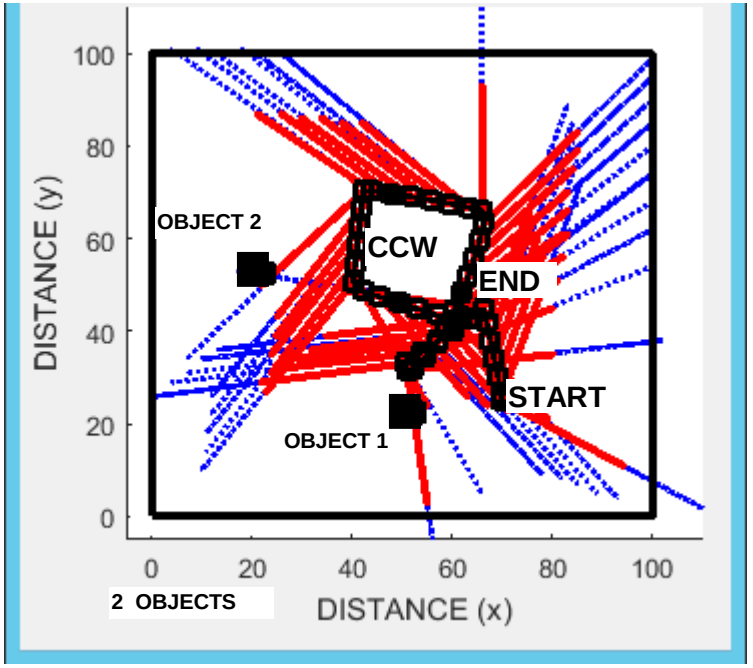
b 

Figure 56. TOURIST AN ON: path simulated for 66 s (21*pi). a. Arena bounded with no internal objects and path shown; b. Behaviors over time and locations in x-y plane. Anticipation moves away from the expected wall, and after the delay, SEEK cues at 6.76 s, yet before 270 deg is turned, at 7.08 s a wall is found by FIND and TOURIST moves towards it, with AHEAD & AVOID again follow the wall. SEEK & FIND are cued once more over 66 s, the same as one cycle for NO AN. SEEK interrupted (every 10th pt. darker) Path: start heading= 0 deg.; Initial: xpos=70, ypos=25, tAVOID=0.125s, tWaitSEEK=5s.



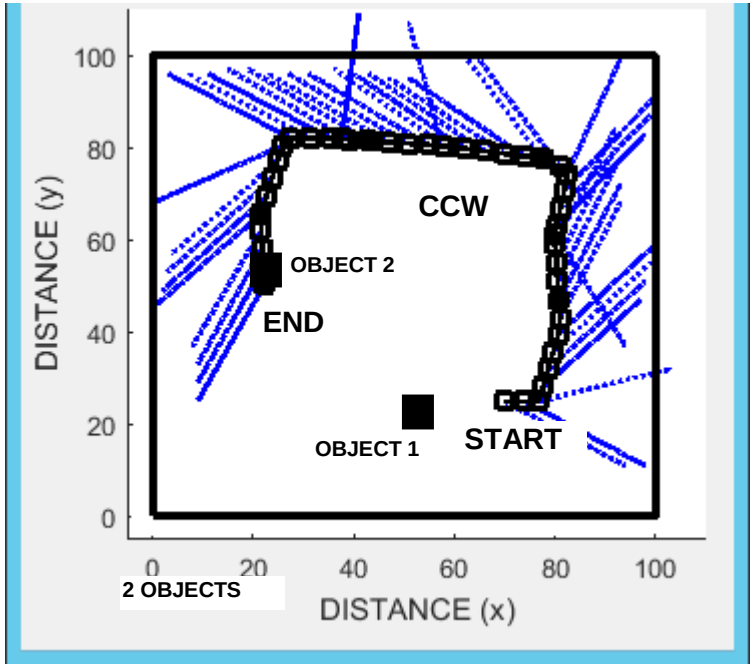
NO ANTICIPATION. 28 cm



ANTICIPATION ON. 28 & 50 cm

Figure 57. AN path simulated for 44 s (14*pi). No Anticipation: 28 cm; (left); No Anticipation on: 28 cm (right). Both initial: xpos=70, ypos=25; When Minimum= Maximum distance= 28 cm, there is effectively No Anticipation; Time shown of 44 s encounters Object 2 with No AN, but passes both Objects with AN On. Dotted lines show path of IR sensor percept beam at each second from the robot location to the IR endpoint 50 cm away, darker line only 28 cm. Random appearing directions occur during the SEEK routine spin that may point in almost any direction, but is only used when AN is On.

SEEK not interrupted (every 10th pt. darker) Path: start heading= 0 deg.; Initial: xpos=varies, ypos=varies, tAVOID=0.125s, tWaitSEEK=5s, NicheLayout=100X100cm, IROffset= 30 deg, speed= 15 cm/s; IRDistMin= 28 cm; IRDistMax= 28 cm.

a. 

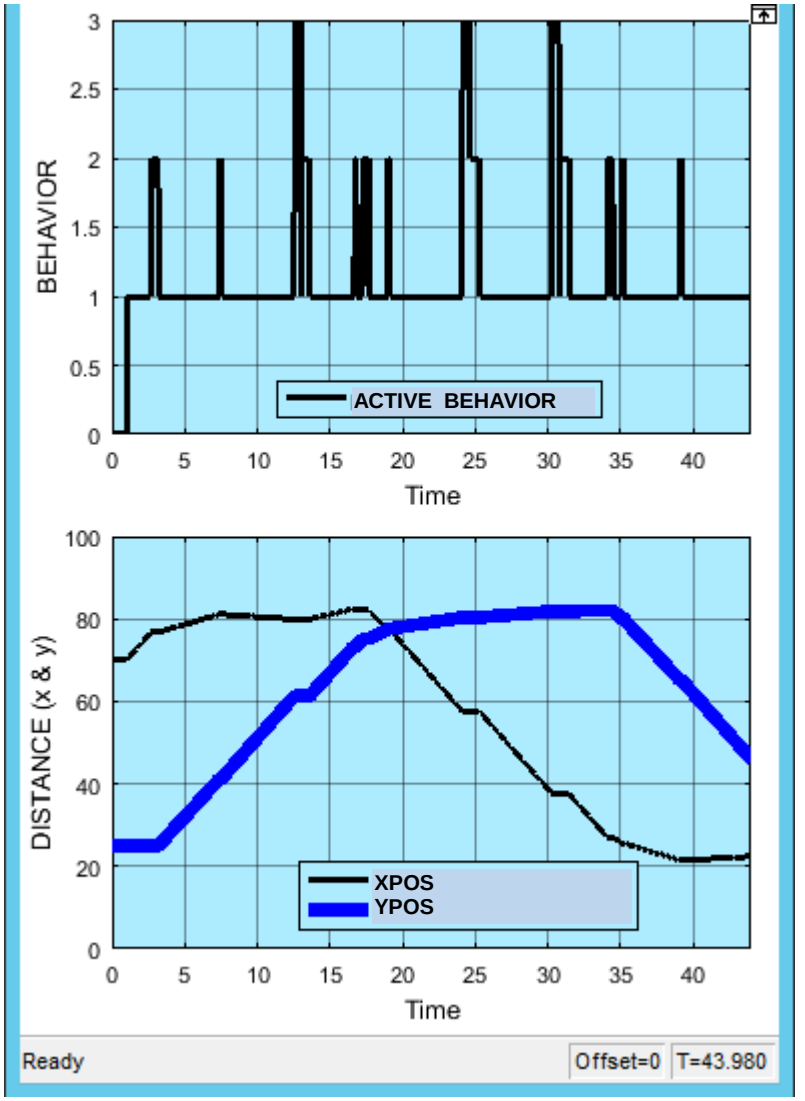
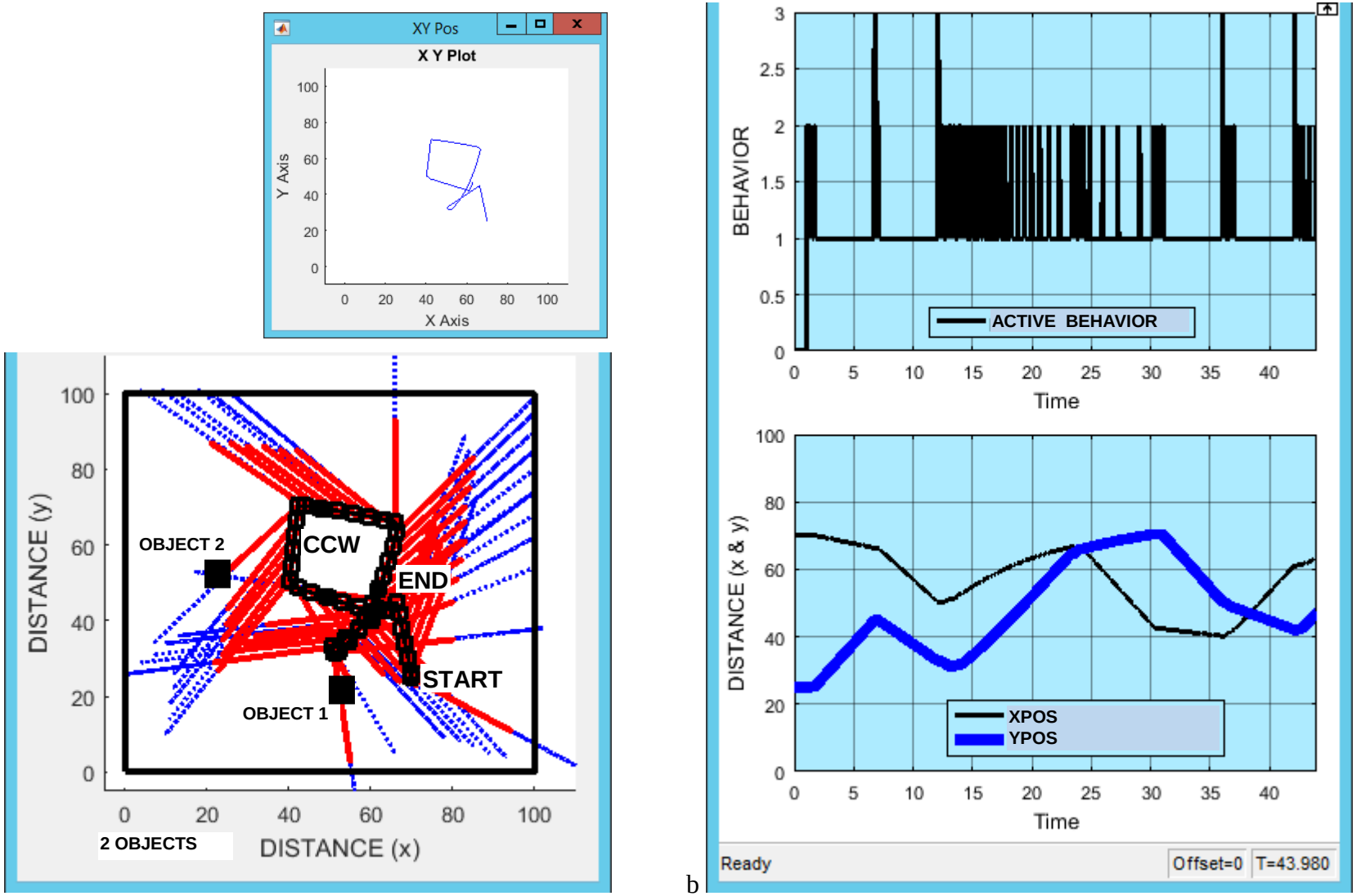
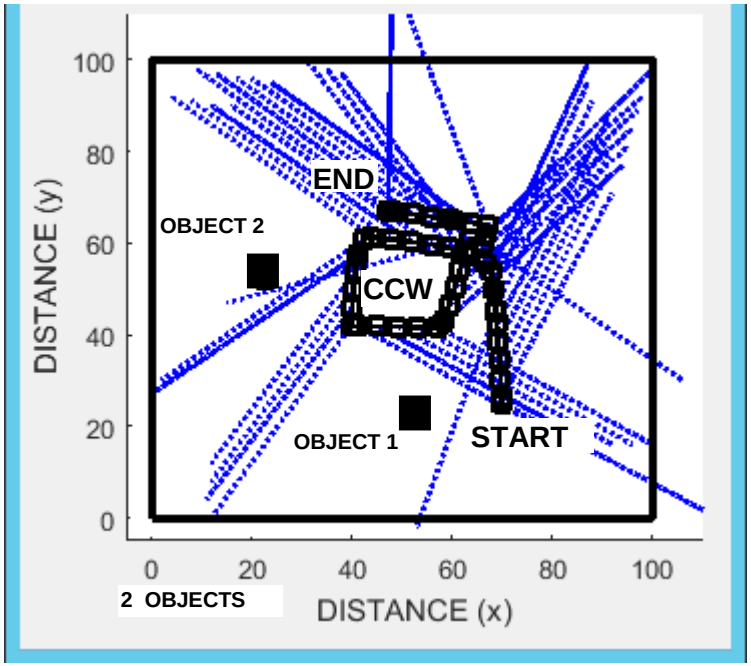
b 

Figure 58. TOURIST NO AN: path simulated for 44 s (14*pi). a. Arena bounded with no internal objects and path shown; b. Behaviors over time and locations in x-y plane. Combined AHEAD & AVOID moves along the expected wall, and after the delay, SEEK cues at about 7 s, and at 2 later times. Error: Agent passes through Object2, as is not possible in the real world. SEEK interrupted (every 10th pt. darker) Path: start heading= 0 deg.; Initial: xpos=70, ypos=25, tAVOID=0.125s, tWaitSEEK=5s.

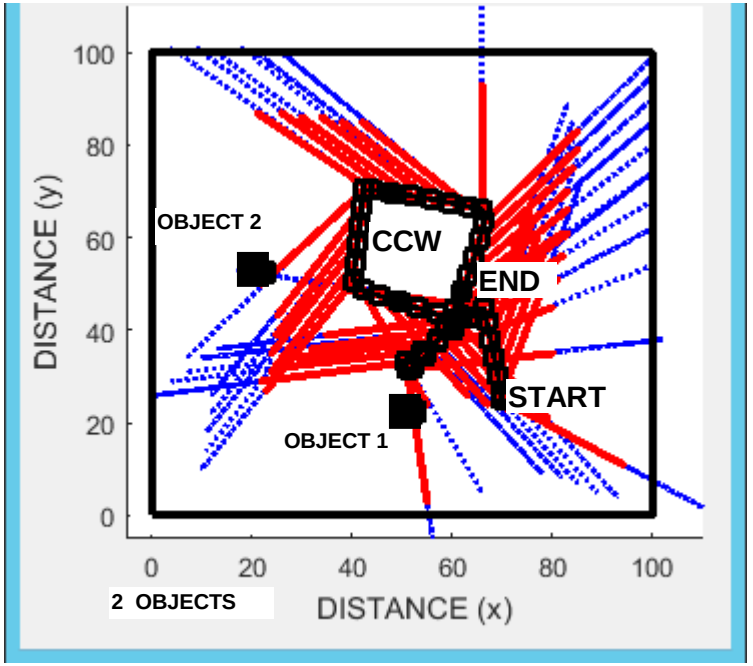


a. b

Figure 59. TOURIST AN ON: path simulated for 44 s (14*pi). a. Arena bounded with no internal objects and path shown; b. Behaviors over time and locations in x-y plane. Combined AHEAD & AVOID moves along the expected wall, and after the delay, SEEK cues at about 7 s, and at 3 later times. Agent passes around both Objects 1 & 2, as desired.
SEEK interrupted (every 10th pt. darker) Path: start heading= 0 deg.; Initial: xpos=70, ypos=25, tAVOID=0.125s, tWaitSEEK=5s.



NO ANTICIPATION. 50 cm



ANTICIPATION ON. 28 & 50 cm

Figure 60. AN path simulated for 44 s (14*pi). No Anticipation: 50 cm (left); No Anticipation on: 28 & 50 cm (right); Both Initial: xpos=70, ypos=25 (right);. Minimum= Maximum distance= 50 cm, there is effectively No Anticipation; Time shown of 44 s encounters no Objects with No AN, and passes both Objects with AN On. Dotted lines show path of IR sensor percept beam at each second from the robot location to the IR endpoint 50 cm away, darker line only 28 cm. Random appearing directions occur during the SEEK routine spin that may point in almost any direction, but is only used when AN is On.

SEEK interrupted (every 10th pt. darker) Path: start heading= 0 deg.; Initial: xpos=varies, ypos=varies, tAVOID=0.125s, tWaitSEEK=5s, NicheLayout=100X100cm, IROffset= 30 deg, speed= 15 cm/s; IRDistMin= 28 cm; IRDistMax= 28 cm

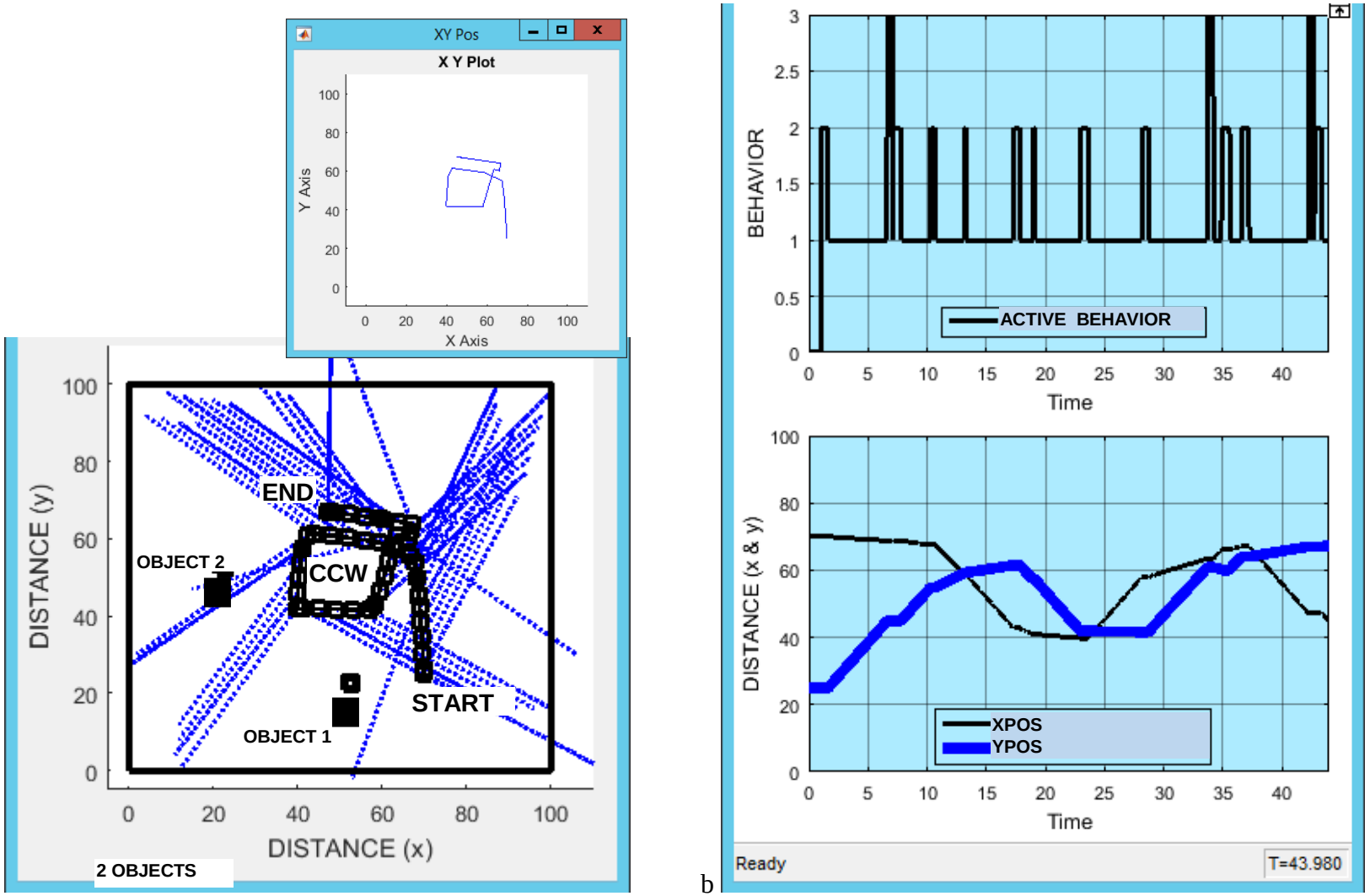
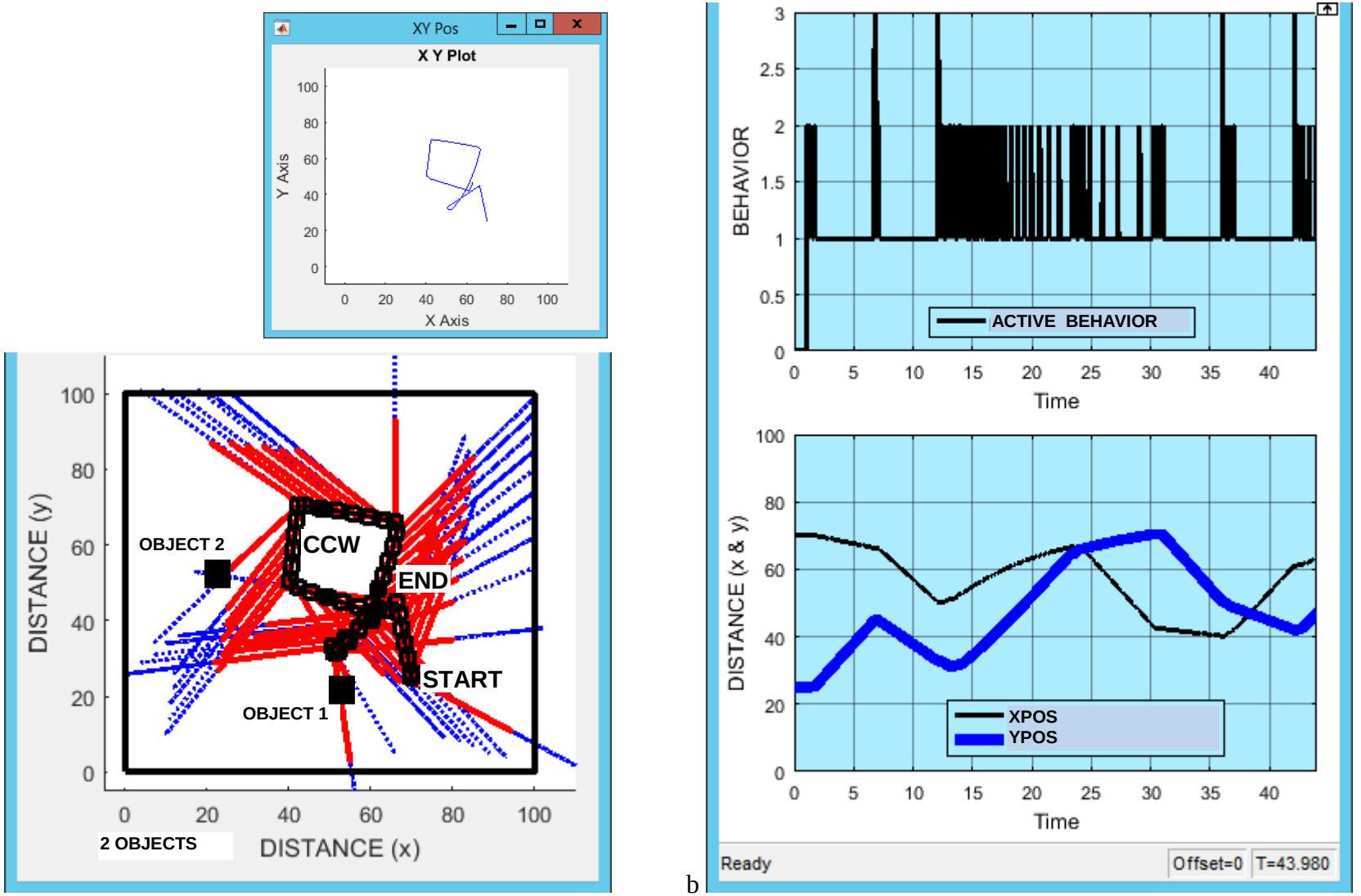


Figure 61Figure 61. TOURIST NO AN: path simulated for 44 s (14*pi). a. Arena bounded with 2 internal objects and path shown; b. Behaviors over time and locations in x-y plane. When Minimum= Maximum distance= 50 cm. Combined AHEAD & AVOID moves along the expected wall, and after the delay, SEEK cues at about 7 s, and at 3 later times. Agent passes around both Objects 1 and 2, as desired in the real world.
SEEK not interrupted (10th pt. darker) Path: start heading= 0 deg.; Initial: xpos=70, ypos=25, tAVOID=0.125s, tWaitSEEK=5s.



a. b

Figure 62. TOURIST AN ON: path simulated for 44 s (14*pi). a. Arena bounded with no internal objects and path shown; b. Behaviors over time and locations in x-y plane. AHEAD & AVOID move along the expected wall, and after the delay, SEEK cues at about 7 s, and at 3 later times. Agent passes around both Objects 1 and 2, as desired.SEEK interrupted (10th pt. darker) Path: start heading= 0 deg.; Initial: xpos=70, ypos=25, tAVOID=0.125s, tWaitSEEK=5s.

For a relatively extended run time for ANSIM, a run with AN ON shows earlier and repeated response to the wall to follow along but not reach a critical distance that is too close to allow the robot to miss the wall (Figs. 50, 51, & 52). Anticipation moves away from the expected wall, and after the delay, SEEK cues at 6.76 s, yet before 270 degrees is fully turned, at 7.08 s a wall is found by FIND and TOURIST moves towards it by a halt of SEEK and initiation of the EXPLORE behavior. Both AHEAD and AVOID again work together to follow the wall. SEEK and FIND are cued only once more over the total run time (66s), which is the same as one cycle for NO AN. Dynamic changes in movement are more gradual and preemptive with Anticipation, or abrupt if Anticipation shows that is needed, as in corners and a new wall encounter. This shows that small adjustments effectively attain task achievement to follow the wall, while avoiding several cues of the SEEK behavior that use larger energy and effort to spin 270 degrees that is considerably more wasteful of resources.

A benefit of anticipation also is shown in these comparisons (Fig. 53). Only a little over one cycle of the arena is made with NO AN, but with AN ON the TOURIST covers the arena twice that much (about 2.5 cycles). Hence, the addition of anticipation allows for a quicker and more thorough coverage of the area, which has advantages in dynamic situations.

For another relatively extended run time for ANSIM, a run with AN ON shows earlier and repeated response to the wall to follow along but not reach a critical distance that is too close to allow the robot to miss the wall (Figs. 54, 55, & 56). The paths of the IR beam were included at one second intervals in these comparisons to more clearly show the range covered by the perceptions by the beam. These illustrate that the repeated cycling with AN ON and the range of

the beam is considerably greater over time than with NO AN. Thus, having anticipation covers a broader range in the same time, and would be able to pick up dynamic changes in the niche more readily than without anticipation.

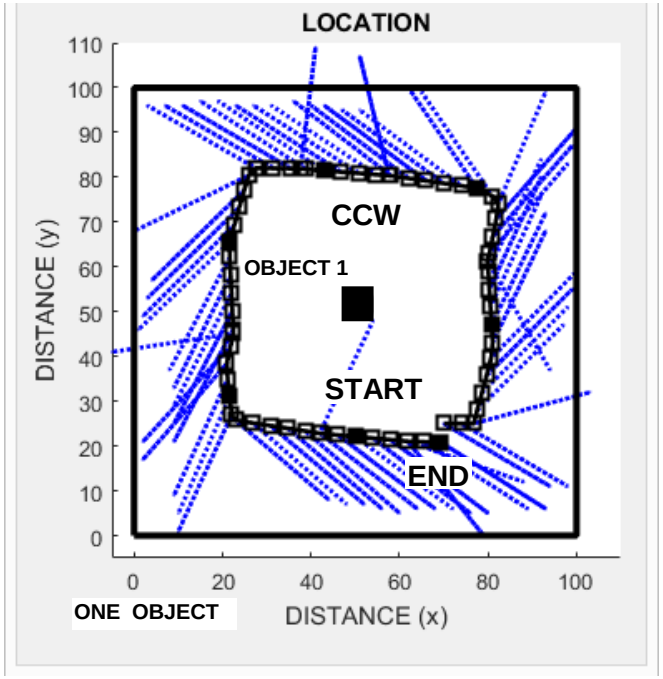
For two objects in the arena and relatively medium length run time for ANSIM, a run with NO AN collides with an object (Object 2) because the IR sensor perceives slightly to the side instead of directly to the front (Figs. 57, 58, & 59). So in this specific case of timing and path traveled, the collision occurs. However, AN ON responds earlier and with more distance from the wall also allows the robot to miss the object. Anticipation appears preemptive for this case, though may partially be a result of the specific setup of the arena with objects.

In addition, for two objects in the arena and relatively medium length run time for ANSIM, a run with a longer IR beam (50 instead of 28 cm) and with NO AN no longer collides with an object but moves a greater distance from the wall that also allows the TOURIST to miss the objects (Figs. 60, 61, & 62). Similarly, with AN ON and the same distances used as before, the TOURIST misses the objects while following the wall, and does so even more smoothly and preemptively than just with the single lengthened distance reference. Thus, adding the anticipation concept makes small adjustments instead of larger ones, so results in less abrupt movements that appear less wasteful of energy and appear to more responses to the objects rather just the arena wall.

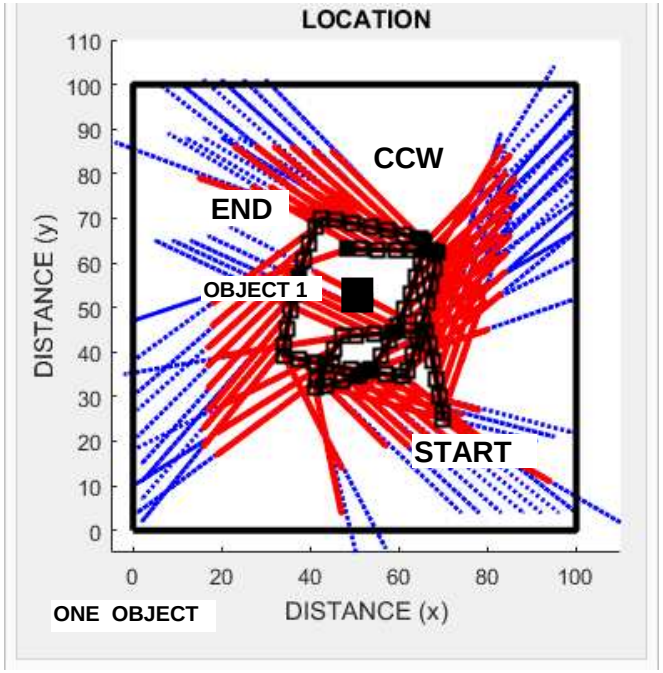
When placing a single object in the center of the arena, different behavior responses are observed for NO AN or for AN ON (Figs. 63, 64, & 65). With NO AN, the TOURIST does not encounter

the object, and only follows the arena wall. When AN is added (AN ON), the object is found during a SEEK spin, and the TOURIST moves towards yet passes by it without any collision. On a second cycle, the results of AHEAD and AVOID along the arena wall do not cue SEEK, so no spin occurs, and thus no encountering of the object. A tighter circle around the center is made, but there is no perception or interaction with the object during the second cycle. Overall, these slightly differing behavior sequences suggest the addition of anticipation allows for greater flexibility in traveling the arena.

When adding two objects in opposing corners, response with AN ON is much more concentrated on the objects (Figs. 66, 67, & 68). With NO AN, the TOURIST only encounters object 2, and uses AVOID to pass round it. With AN ON, the TOURIST passes by both objects in multiple cycles around the arena. Thus, with AN ON, the TOURIST is more responsive to the objects in the arena than without AN. Both objects are anticipated, and AHEAD & AVOID move away from them before any destructive encounter. Location of the objects in relation to arena walls does affect the outcome, and cued behaviors operate before any negative encounters occur for the simulations, as was expected.



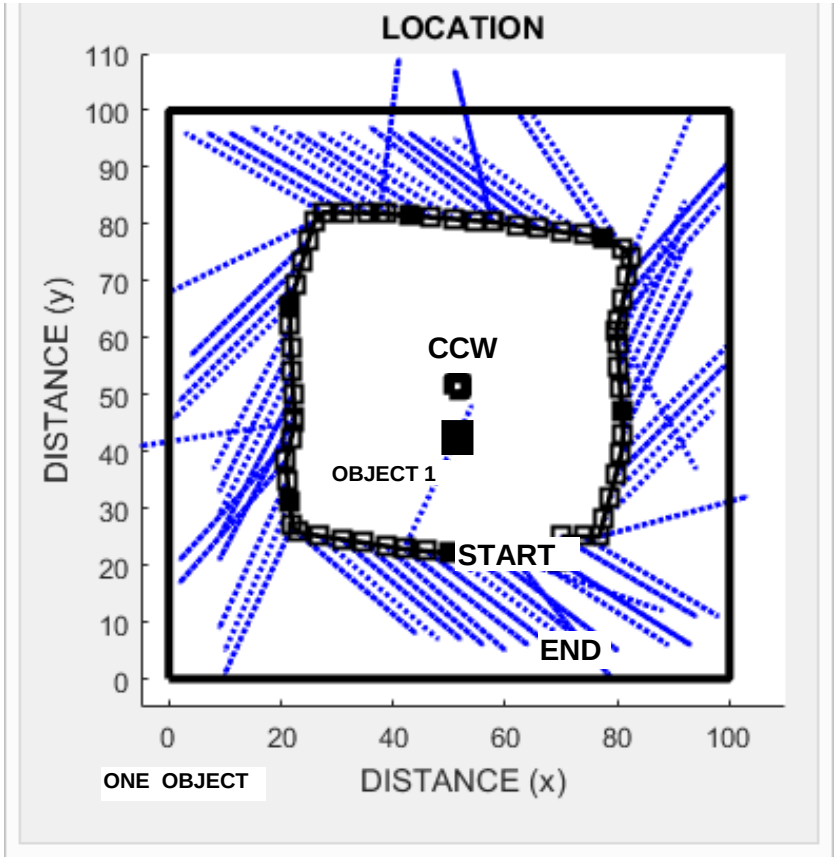
NO ANTICIPATION. 28 cm



ANTICIPATION ON. 28 & 50 cm

Figure 63. AN path simulated for 66 s (21*pi). No Anticipation: 28 cm (left); Anticipation On: 28 & 50 cm (right); Both Initial: xpos=70, ypos=25 (right); For Minimum= Maximum distance= 28 cm, there is effectively No Anticipation; **Time shown of 66 s encounters no object with No AN, and passes by the one object with AN On.** Dotted lines show path of IR sensor percept beam at each second from the robot location to the IR endpoint 50 cm away, darker line only 28 cm. Random appearing directions occur during the SEEK routine spin that may point in almost any direction, but is only used when AN in On.
SEEK not interrupted (every 10th pt. darker) Path: start heading= 0 deg.; Initial: xpos=varies, ypos=varies, tAVOID=0.125s, tWaitSEEK=5s, NicheLayout=100X100cm, IROffset= 30 deg, speed= 15 cm/s; IRDistMin= 28 cm; IRDistMax= 28 cm

a.



b

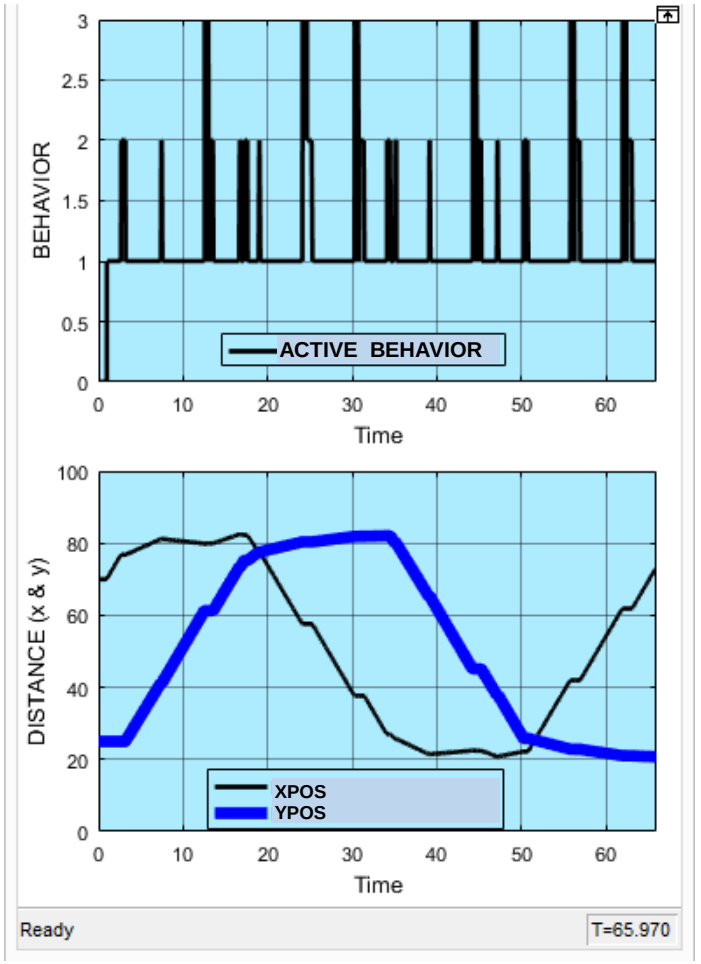
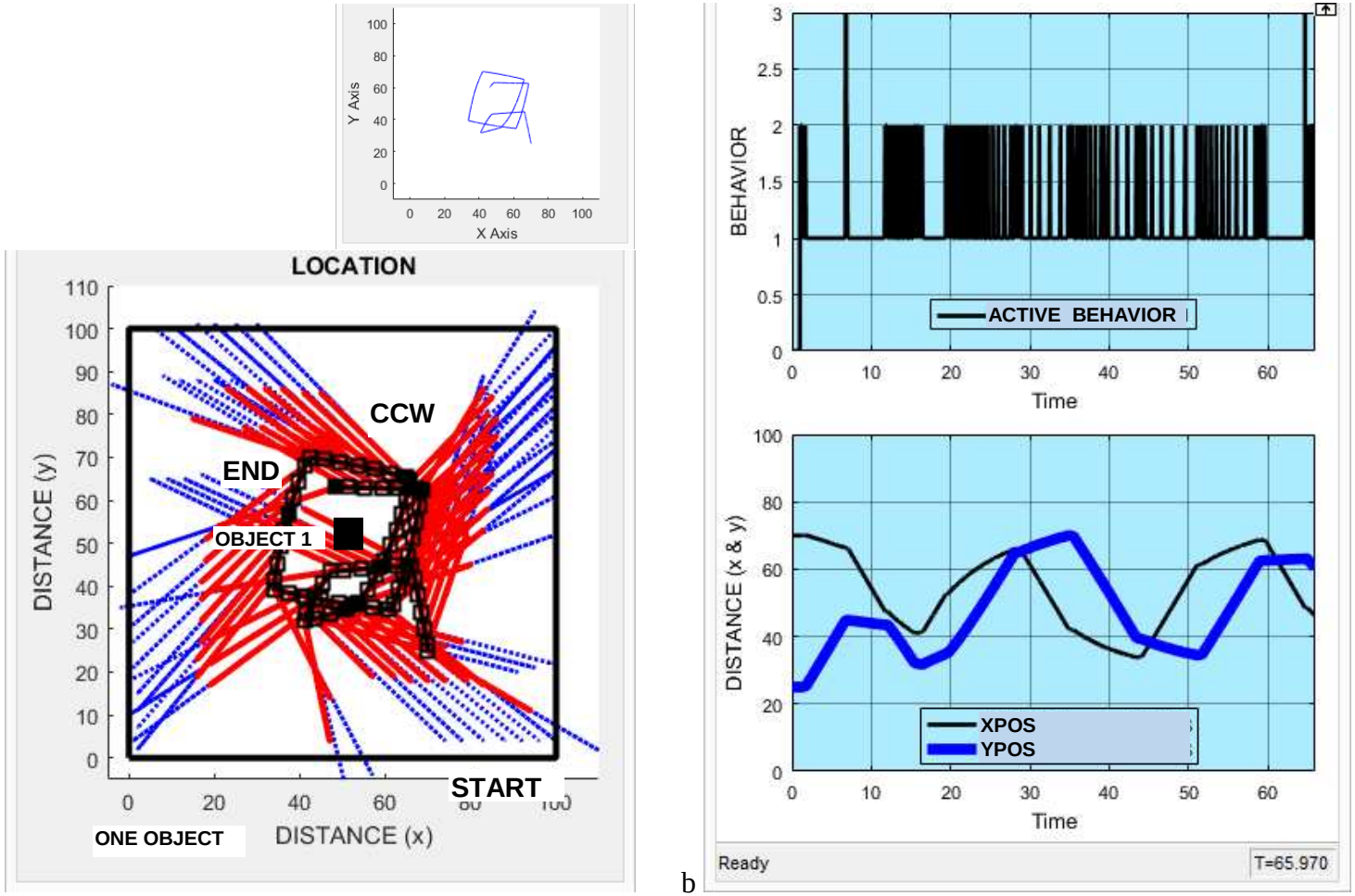
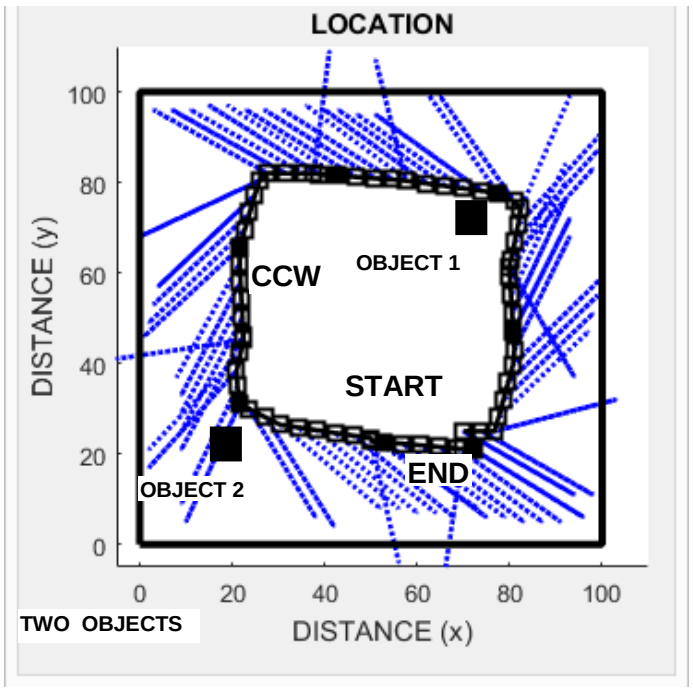


Figure 64. TOURIST No AN: path simulated for 66 s (21*pi). a. Arena bounded with no internal objects and path shown; b. Behaviors over time and locations in x-y plane. Combined AHEAD & AVOID moves along the expected wall, and after the delay, SEEK cues at about 13 s, and at 5 later times.
SEEK interrupted (every 10th pt. darker) Path: start heading= 0 deg.; Initial: xpos=70, ypos=25, tAVOID=0.125s, tWaitSEEK=5s,

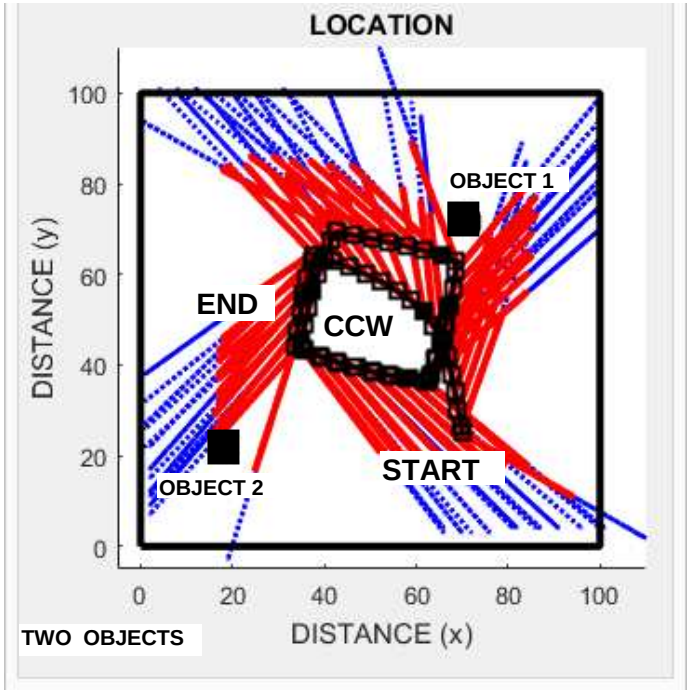


a. b

Figure 65. TOURIST AN ON: path simulated for 66 s (21*pi). a. Arena bounded with no internal objects and path shown; b. Behaviors over time and locations in x-y plane. Anticipation moves away from the expected wall, and after the delay, SEEK cues at 6.76 s, yet before 270 deg is turned, at 7.08 s an object is found and the TOURIST moves towards it, combining with AHEAD & AVOID to again move near the object. One more SEEK & FIND behavior combination is cued over the interval shown. The insert small graph shows the path more clearly without indicated IR beam (upper left).
SEEK interrupted (every 10th pt. darker) Path: start heading= 0 deg.; Initial: xpos=70, ypos=25, tAVOID=0.125s, tWaitSEEK=5s.

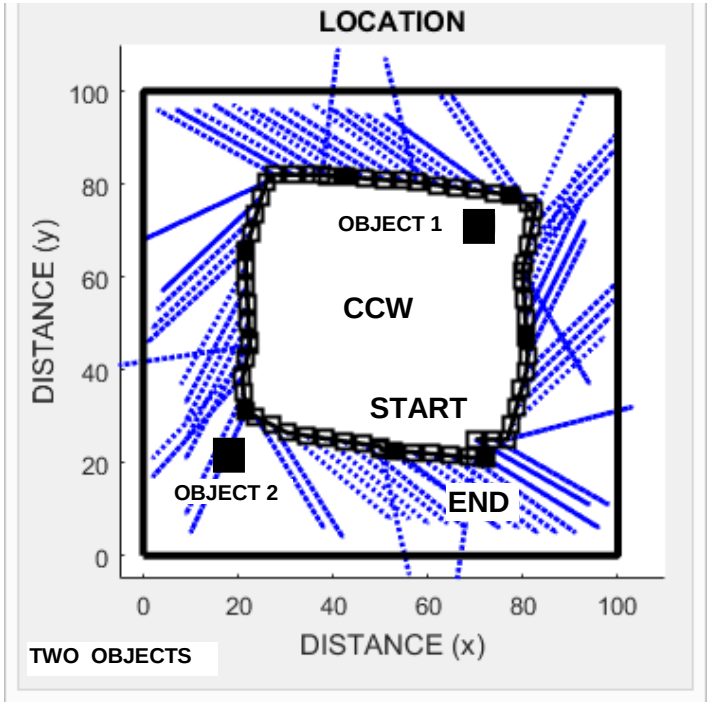


NO ANTICIPATION. 28 cm



ANTICIPATION ON. 28 & 50 cm

Figure 66. AN path simulated for 66 s (21*pi). No Anticipation: 28 cm (left); Anticipation On: 28 & 50 cm (right); Both Initial: xpos=70, ypos=25 (right); For Minimum= Maximum distance= 28 cm, there is effectively No Anticipation; Time shown of 66 s **encounters only object 2 with No AN, and passes by both objects with AN On. With AN ON, the TOURIST is more responsive to the objects in the arena than without AN.** Dotted lines show path of IR sensor percept beam at each second from the robot location to the IR endpoint 50 cm away, darker line only 28 cm. Random appearing directions occur during the SEEK routine spin that may point in almost any direction, but is only used when AN in On.
SEEK not interrupted (every 10th pt. darker) Path: start heading= 0 deg.; Initial: xpos=varies, ypos=varies, tAVOID=0.125s, tWaitSEEK=5s, NicheLayout=100X100cm, IROffset= 30 deg, speed= 15 cm/s; IRDistMin= 28 cm; IRDistMax= 28 cm

a. 

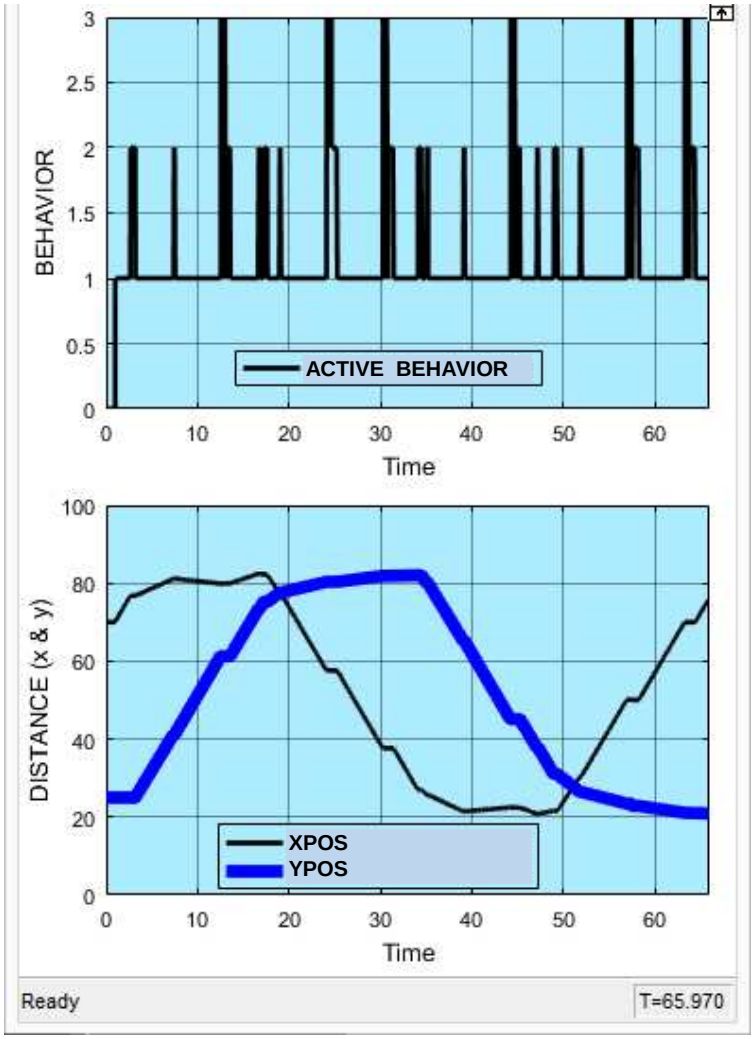
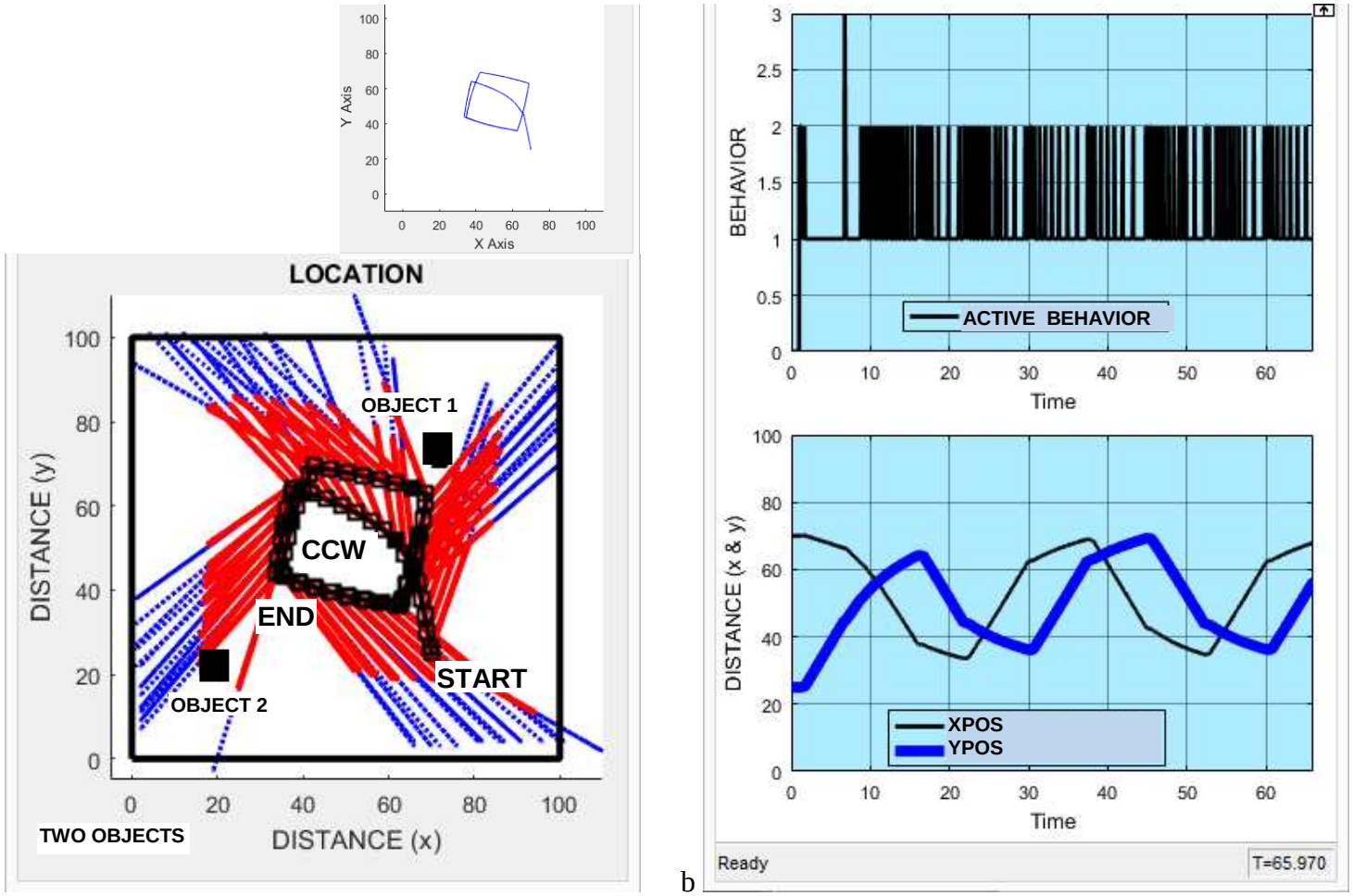
b 

Figure 67. TOURIST No AN: path simulated for 66 s (21*pi). a. Arena bounded with two internal objects and path shown; b. Behaviors over time and locations in x-y plane. Combined AHEAD & AVOID moves along the expected wall, and after the delay, SEEK cues at about 13s, and 5 later times. The TOURIST goes around object 2 when it is encountered. SEEK not interrupted (every 10th pt. darker) Path: start heading= 0 deg.; Initial: xpos=70, ypos=25, tAVOID=0.125s, tWaitSEEK=5s.



a. b

Figure 68. TOURIST AN ON: path simulated for 66 s (21*pi). a. Arena bounded with two internal objects and path shown; b. Behaviors over time and locations in x-y plane. Anticipation moves away from the expected wall & objects. SEEK cues at 6.76 s, yet before 270 deg is turned, at 7.08 s a wall is found and the TOURIST moves towards it. Both objects are anticipated and AHEAD & AVOID move away from them before any destructive encounter. No more SEEK & FIND are cued over the period, since objects are readily responded to. Insert graph shows the path more clearly (upper left). SEEK interrupted (10th pt. darker) Path: start heading= 0 deg.; Initial: xpos=70, ypos=25, tAVOID=0.125s, tWaitSEEK=5s,

ANTICIPATION BENEFITS

Simulations above illustrate how adding anticipation to the TOURIST robot agent by infusing the system dynamics of improved archetypes demonstrates the benefits of including anticipation in a simple robotic system. First, the system performance in the arena was verified to be that expected for the previous wall following robot from the work of Connell (1990), both traveling along the wall and avoiding objects using the three behaviors EXPLORE, AVOID, and SEEK. Then anticipation (AN) was added via the two system archetypes: Limits To Growth (LTG) and Shifting The Burden (STB). Since a key trait of AN is to observe agent behavior *before* the outcome of the situation is certain, the illustration showing small adjustments to the AN path prior to encountering the wall or objects is evidence of AN working in the altered system. In essence, the system appears to *know the future* of later interaction with the wall or object, and makes the adjustments prior to that. The system dynamics to the archetypes working within the system structure manifests the behavior choice that best matches the niche condition.

In biological systems, an organism may need to practice to develop the quick response behavior to the perceived niche conditions. Similarly, in artificial robotic systems the designer builds in the response behavior that matches the expected niche conditions. A relatively small set of likely conditions must be identified by the designer, and the direct desired behavior is matched to the niche conditions. This choice to manifest the specific behavior is based on some method to calculate a fitness or suitability of the current behavior to match the niche. In general, a weighted combination of several factors can be combined to determine such a suitable fitness value. For the TOURIST, distance to a wall or object is perceived by the infrared (IR) sensor, and a ratio of the distance to a critical minimum distance is used to form a relative scaled value that allows

execution of the choice behavior. This application of anticipation dictates use of the AHEAD behavior to cue the AVOID behavior more often, yet for briefer time periods, and as needed for smaller path adjustments that move away from the wall *before* it is encountered. Effectively, with AN the SEEK behavior occurs less often (about only a third as much), and thus keeps from using large amounts of energy associated with the 270 degree CCW spin. Considerably smaller adjustments, and much fewer larger adjustments, represent both an energy savings, and coverage of a relatively larger area in a repeated manner. The increased coverage enables more rapid and effective identification of objects and potential dynamic changes within the arena. In this way, **behavior choice is effectively determined by an expected future condition, that of encountering a wall or object**. Varying locations of the objects in relation to arena walls does affect observed outcomes, yet cued behaviors operate before any negative encounters occur for the simulations, as was expected.

The simulation contains models of both the agent itself and the niche environment. The model keeps track of the location and traveled path, as well as the heading for the next forward travel. Current and past behaviors are tracked, too. The simulation contains a simple description map of the initial layout of the arena and objects. Distance from the robot agent to the arena boundaries is perceived within desired ranges, so the situated location of the robot in relation to other items in the arena is also known, at least in the forward looking direction (with a slight angle offset). These parameters correlate the robot agent with the niche and resulting behavior, so there is a connectedness with the niche world, though the perceptions are minimal. Hence, the robot appears to be synchronized in the arena as it moves dynamically in the overall system arena. Observing these combinations of behavior for the artificial robot arena system, similar to what

might be observed for a biological organism in an arena, indicates that the overall robot with its niche world has the elements of anticipation. Especially, the most important elements to observe are that the system manifests behavior *before* the outcome is certain, so that observed behavior *appears to have known the future* outcome before it occurs.

ROBOT INSTANCE OF ANTICIPATION

The TOURIST robot was fashioned after the example of the wall follower (e.g., MURAMATOR, meaning 'wall follower' in Latin) designed and built by Connell (1990) using a colony architecture as a revised subsumption architecture after that developed by Brooks (1999). The TOURIST originally contained the three behaviors (EXPLORE, AVOID, and SEEK) as used by Connell to exhibit behavior-based robotics reactive response to a simple bounded arena and to objects placed in the arena. The TOURIST robot was observed to perform to verify the behaviors as expected for desired task achievement, to follow the wall and avoid objects to protect from collisions and destruction of the robot physical structure. The embedded microprocessor contained a Processing code that was ported to a computer simulation, ANSIM (for ANticipation SIMulation) in SIMULINK with underlying behaviors translated into a subsystems in MATLAB representation. A virtual niche environment was constructed as an arena with objects for the robot to virtually travel within. A virtual sensing procedure was developed to mimic the operation of an infrared (IR) sensor, to perceive and form a percept of the niche conditions at any set time. Initial tested results ensured the expected operation of the virtual robot in the arena, using graphs of the sequence of behaviors and the path traveled in the arena. These steps allowed for the ANSIM model to depict robot response using a reactive behavior-based robotics approach to navigate in a simple niche.

The notion of anticipation (AN) was added to the TOURIST using the system dynamics of the archetypes Limits to Growth (LTG) and Shifting The Burden (STB). The LTG archetype includes asymptotic approach to some value based on a system constraint. The first LTG archetype was added to the ANSIM model by adding an AHEAD behavior routine that used a further perception of distance to objects and walls, and scaled the turn aspect of the AVOID behavior to make relative small adjustments in a ratio of perceived object distance to that of some minimum distance for a safe turn to miss the object or wall. The LTG archetype is considered appropriate for this response since it sets asymptotic limits to results realized by the AVOID behavior. The second STB archetype was added to the ANSIM model by adding a FIND behavior routine that interrupts the spin portion of the SEEK behavior if some object or wall is perceived at a further distance in a prescribed range. This invokes the TOURIST to stop SEEK behavior and start the EXPLORE behavior to move toward any object or wall. The STB archetype is considered appropriate here since it replaces the symptomatic solution as the SEEK spin with a fundamental solution to approach an object or follow a wall that is perceived nearby. Simulations of various scenarios show the effects of the archetypes to add anticipation to the robot navigation. Anticipation enables small adjustments before any critical approach distance is reached to a wall or object. Changes are less abrupt, and the cumulative effect is to move along the wall or pass objects without any contact. Fewer cues of the SEEK behavior (only a third as many) keeps from wasting relatively large amounts of energy merely to spin in place as the SEEK behavior to reach a new heading. Also, by using various maximum distances to use with anticipation on, very little effect was seen if there is a zero range of distances used, increasing to a useful level that anticipated walls and moved before contacts, to a longer-range detection that was actually too excessive. The excess case is over-anticipation, and reacts so prematurely as to

create turning that is not productive, never approaching a wall or object. Thus, application of methods to enable anticipation can actually be overdone to the point of deleterious effects. On a more positive note, striking a balance between a minimum (28 cm) and maximum (50 cm) distance for turn scaling can produce a useful range that demonstrates the traits of anticipation.

Currently, the abstracted ANSIM inferences in the formal system have been decoded back into the NS of the physical TOURIST robot, overcoming the challenges of making SIMULINK code into the Processing code used on the robot microprocessor (i.e., Arduino Nano). Initially, microprocessing code on the TOURIST robot used to manifest behavior choice was embedded in only about 40 lines of Processing code, and was a key portion of the code translated by encoding to SIMULINK/MATLAB subsystems to enact the behaviors correctly in simulation by ANSIM. Actually, this was the easiest part of the ANSIM simulation to enact, since the more difficult programing effort was to represent the surrounding arena environment, and the percepts of the local niche that manifest behavior. Once proper behavior operation was verified in simulation (behaviors resulted as expected), anticipation was added with a switch variable (AN01) that allowed simulation either without or with anticipation archetype system dynamics behaviors. Working simulation added code was decoded back to the Processing language for the microprocessor operation on the TOURIST robot. Effectively this was about 20 lines of code in two different subsystems. A minimalist perspective aligns well here with the idea that a significant change was possible with small alterations in operation of the actual physical TOURIST robot.

Several demonstration runs of the TOURIST robot showed the effect of decoding the simulation FS to the robot NS. Setting the switch to select operation without anticipation (NO AN) resulted

in behaviors like those observed originally in the robot before simulations, with expected wall following and object avoidance. Thus, it was verified the same behaviors were observed for the TOURIST after the NS was encoded to the FS, and also after the FS was decoded from simulation back to the NS. Video records of the runs showed successful operation most of the time, but occasional collisions with walls occurred for certain specific angles that the wall follower was not able to effectively detect the wall to be avoided and followed (Fig. 69). Thus, AVOID behavior is limited by the angle setting of the IR beam, yet that angle is necessary to enable the wall following task achievement. The SEEK behavior similarly would lead to wall collisions if the angle of approach after the resulting spin was undesirable. Thus, both successes and limitations of the original TOURIST robot performance for task achievement was validated to occur after decoding back to the robot NS.

Anticipation was added by setting the switch to activate anticipation behaviors (AHEAD and FIND). AHEAD was added to the selection for initiating and scaling timing of the AVOID behavior, while FIND permitted interruption of the SEEK behavior to engage new nearby walls and objects. With anticipation on (AN ON), the TOURIST robot made earlier smaller adjustments to the upcoming walls or objects using AHEAD and AVOID behaviors, staying farther away from them than was observed when there was no anticipation (NO AN). When SEEK was invoked, which was less often, FIND was shown to interrupt the spin as expected and move towards (yet AVOID) nearby walls and objects, as shown by video record (Fig. 70). These near walls and objects would not have been detected without anticipation, since the original spin of the wall follower was a timed procedure process that was not interrupted until it was complete. Occasionally the robot would collide with the wall at a certain unfortunate angle, yet

generally the workings of SEEK and FIND would move away successfully to continue EXPLORING of the arena. Thus, with anticipation on (AN ON), the TOURIST robot moved more smoothly along walls and more readily located walls and objects to approach, yet had fewer collisions, and resolved them more adequately than occurred without anticipation.

ANTICIPATION METRIC

A specific metric (ANNum) was developed for operation of the TOURIST robot considering the distance measured by the IR beam and the area covered by the motion of the robot at any time instance (Appendix A4). The metric (ANNum) can be used to compare travel of robot paths from ANSIM model for a set time and arena configuration. Area covered by the IR beam during each occurrance of behavior for EXPLORE, AVOID, and SEEK/AHEAD (anticipation on only) reflected impulse response to the niche conditions and were accumulated over time. Greater metric values were found with anticipation on, reflecting more behavior responsiveness to the niche per unit time when anticipation was used.

Figure 69. TOURIST robot video with no anticipation (NO AN). The TOURIST robot occasionally collides with and is stuck at a wall.

Figure 70. TOURIST robot video with anticipation on (AN ON). The TOURIST robot successfully escapes from a collision with the wall.

CHAPTER 5. SUMMARY

OVERVIEW

Notion of Anticipation

Anticipation is considered to have benefits beyond mere reaction to present niche conditions. The notion of anticipation presented here assumes predetermined choices exist that can be used to manifest behavior for certain expected conditions, yet concedes a logical reasonable response may not be available for uncertain and previously unknown and unconsidered conditions. Since a general purpose robot is not considered to exist, instead a robot agent with specific task achievement, or purpose, is considered to have a known set of behaviors that match a set or repertoire of expected conditions in a specific defined niche environment. The goal for future work with anticipation is to define the specific desired behaviors that match the expected niche conditions and result in desired task achievement. Encountering unexpected niche conditions should result in safe behavior that does not appear illogical or unreasonable from a human perspective, and may be as simple as a noncommittal behavior that waits for the next known expected condition to occur.

Anticipation Traits

The major benefit of anticipation may be a more timely reaction to change that occurs in the niche. The main trait of anticipation is that action occurs *before* an expected outcome is certain, so quicker action should suggest the *agent knew what was about to happen* all along. This trait has advantages across many areas where conditions are likely to change, albeit within a range of

standard values that are matched to an appropriate behavior response. On a simple daily basis, this is akin to using robot agents to perform menial tasks on timers to make coffee or start a routine process such as closing blinds on windows and reducing temperatures for the night. Scheduling the movement of trains, such as those at an airport (e.g., Denver International Airport, CO), is a larger and more impactful procedure seen by many people. These simple items we already view as automation, and are commonplace. But anticipation has the additional requirement, that of acting *before* an outcome is certain. Although current automation processes work well in a sequence of scheduled events, such as train schedules, anticipation has the trait of including behaviors that are matched to niche conditions that are expected, yet not know when, where, and how they might be occurring. Anticipation has a type of flexibility to deal with a wider range of possible outcomes, and yet behave in a way that appears to have *known the final outcome* all along. This requires definition of a specific anticipation set repertoire, though it may still be small in size, yet enables quick and decisive action when conditions favor a certain expected outcome. Unfortunately, the early reaction could have adverse effects if a slight change in niche conditions actually favors a different outcome than the chosen behavior. Thus, anticipation also requires the ability to make behavior change rapidly and repeatedly with those changes found in the niche.

Anticipation Benefits

Considering these traits, anticipation has significant benefits due to its quick choice of a behavior and ability to switch to another more favored behavior. This should provide more rapid response than occurs for a strictly reactive system, and produce outcomes that better fit the dynamic changing current niche. The more decisive anticipatory behavior may be more explosive in

nature to create a stronger impact as perceived by an observer. The behavior arises from a model of the existing world view that determines a certain response is most likely favorable and rewarded, and thus make choice of the best matched behavior.

Anticipation should enable smoother and more effective behavior choices. Wait and indecisive time should be reduced, while effective action should flow from step to step fluently without glitches or failures. Less problems should be observed in general. But more importantly, a series of successful behaviors should repeatedly result in desired task behavior in both the short and long run. The preconceived behaviors that are matched to expected niche conditions are endowed in the system structure by the designer through previous understanding of the likely situations, understanding and preparing for likely system archetypes flaws and acceptable solutions, and building in the correct most favorable responses. This requires a designer or system constructor to fully understand the niche environment and potential behavior outcomes that are desired to be reached to provide task achievement.

Anticipation in Robotics

Anticipation and its operation applies to robotics specifically in at least two separate ways. First, it requires the designer to fully describe expected goals and operation of the system within constraints, and to produce the robot structure and process responses in software programming that can sufficiently attain those goals and operation. Secondly, the autonomous operation of the robot must properly acquire percepts of the niche to allow the correct matched selection of a choice behavior, and to do so autonomously and repeatedly as long as it has sufficient power to

operate. Granted, a mobile robot has an obvious constraint of limited energy supply that eventually restricts ongoing successful operation.

Anticipation is not able to avoid all possible negative situations that might occur for a robotic system. Indeed, the early action before the outcome is known might actually choose the wrong behavior, or one less suited, if the conditions in the environment unfold differently than the local area of the niche indicated. Recall the niche is the area local to the robot that is sensed. Conditions slightly outside that perceive niche may differ, or the percept might be erroneous due to unexpected conditions in the larger environment, something as simple as bright light or shadows. It has been observed in testing that bright light that contains a high level of IR can actually be perceived as an object, when in reality there is no nearby object. To counteract a premature behavior choice, the robot system must continually sample the niche often enough to permit a subsequent change in behabior that makes the desired adjustment that truly matches the actual niche. Therefore, the robot 'catches' itself in a possible incorrect choice, and makes a proper adjustment, on that will allow for overall task achievement in the niche.

Overall, anticipation must go beyond a robot merely reacting to an object, but should combine percepts of the niche to choose behavior that smoothly and gradually moves to a desired outcome, rather than having to make last second abrupt changes that are uncertain and may require more energy and resources to immediately solve the problem or perform an operation. The result should allow the robot agent to act in a way that an observer would say the agent *already knows the outcome* ahead of time, *before* the outcome is actually certain.

CONCLUSIONS

Insights and Contributions

This work contributes to the field of robotics with the following findings:

1. A robot architecture that includes anticipation performs more smoothly and preemptively to make choice of behavior to better fit the niche condition and attain desired task achievement.
2. The congruence framework assists in engineering a robot to perform in the natural system based on correctly decoding inferences simulated and shown successful in an abstracted formal system.
3. Two known system archetypes can readily be coded into a formal system model to affect system operation: Limits to Growth, and Shifting of the Burden (or Goals). Simulation of the system archetypes allows system dynamics to be understood and documented, and allows possible solutions to known problem archetype results to be understood and modified for desired problem solutions.
4. Applying the simulation to operation of a TOURIST robot with anticipation built into the archetype programming illustrates the advantages of including the notion of anticipation. The anticipation methods allow a TOURIST robot agent to travel a smoother path and make choice of small increments in behavior change that produce more desired longer term responses. With anticipation, numerous small adjustments are made that require less energy than large spins of the SEEK behavior, so only one third of the SEEK behaviors occur, and thus waste less energy and time. Also with anticipation, the TOURIST makes twice as many cycles of the area at the same speed and in the same time, so a broader range of area is covered and can more readily perceive any dynamic changes in the overall arena. The ANNum metric as developed for describing operation of the TOURIST robot showed greater

metric values were found with anticipation on, reflecting more behavior responsiveness to the niche per unit time when anticipation was used.

5. The path traveled by the TOURIST with anticipation appears to know in advance the presence of objects or walls, and to undertake a choice of behavior to avoid negative contact and move along the wall in a desired fashion for task achievement, and to do so in a way that the robot appeared to *know the outcome before it was certain to occur*.

FUTURE WORK

Anticipation Application

It is ironic to discuss future work regarding anticipation, since anticipation is always about future expectations. The task before us now is to extend anticipation of robotics into other currently unused and untested areas, and especially those with shorter cycle times. A fundamental example is that of self-driving vehicles, one where much classical robotics has so far been applied to attempt to overcome the many complicated uncertainties in a driving environment that has been created for humans. Though a human in the loop (HIL) approach may be laudable and prudent from a safety standpoint, the more likely approach to be successful should craft the niche to align with needs of a robotic agent using current affordable technology. Anticipation added to driverless vehicles would reduce response time and provide smoother operation.

Operational Description

Overall, anticipation is used to define a repertoire of desired behaviors that match a specific niche setting, such as movement of busses or similar transports in a known corridor. This

approach is actually already in place for subway trains, with humans routinely vigorously forbidden from the domain of the vehicle. So instead of attempting to design a driverless vehicle for an innumerable n-p complete set of solutions, the niche should be better defined for the agent itself, to make the repertoire of reasonable behaviors finite and manageable. So part of the application of anticipation to control and robotics is to establish a known environment where niche conditions are readily known, and proper behavior responses can be matched to the current niche.

Instances In Agriculture

As an example instance, environment redesign for anticipation already exists in agricultural systems at some basic levels. Most obvious, agricultural equipment is built to standard width and row sizes to accommodate all processes from planting, to cultivation, to harvesting, eventual processing, and preparation for the succeeding crop. The choices for plant behaviors favors the ones desired for human benefit, and allows for the greatest or at least some measured level of reward for the efforts and costs incurred. Anticipation is incorporated in the plants themselves by selecting species and indeed cultivars that make it readily possible to enlist aid of machinery for the processes mentioned above. Unfortunately, conflicting views may arise, such as needing firm fruit for transport at the expense of flavor quality. Such tradeoffs are part of the notion of anticipation, since managers and operators must make real choices as to what traits are acceptable for at least a marginable number of individuals to favor and use the product. Anticipation is considered a benefit to add to future robotic machine designs, especially those for improving culture of intensely grown plants (e.g., hops for flavoring) or for improvement of menial harvesting procedures (e.g., strawberries that are still picked by human hands). The niche

must be sufficiently and properly understood to allow for the robot agent to select specific behaviors to successfully achieve desired tasks.

Instance In Space Exploration

On a broader scale, anticipation can be incorporated into such far-reaching areas as space exploration. Indeed, calculations of planet movements already are used to determine locations for possible and more importantly reasonable goals of space travel and exploration. Physical factors of gravity and inertia are included to anticipate what may be an acceptable payload, and we must take along enough of our own air to make a safe return trip. In a more refined way, anticipation by robotic agents can be enlisted to undertake maintenance schedules for living quarters both inside and out, while making sure food preparation and waste disposal are most pleasurable or tolerable for participants. This may be especially vital as we begin to undertake recreational trips into space. Robots may take on the likely servant role, or be devised specifically for entertainment, and in such cases using anticipation to meet both our physical comfort needs (e.g., food and drink) as well as interact with us in ways that are entertaining for most people (e.g., social interactions akin to games with appropriate challenges, and physical exercise). Anticipative ordering, as is already patented on earth for the Internet, might be commonplace. Robotic decisions might be enlisted to provide for human safety, making hard but reasonable choices when resources are limited, and return to a safer venue may be in order. Though we are reluctant to agree that actions by the robot or machine are to our benefit, we still respond to warning lights in our car, and let traffic lights dictate our daily travel routines. This requires that some operator and designer of the overall systems is constrained to build in anticipated choices that can ensure successful task achievement and completion.

Ideas Related to Anticipation

There are many related areas of study to undertake. The selection of anticipation as a topic for study arose from consideration of all elements that might make a system robust. This exercise resulted in a 'robustness orbit' diagram, having a variety of 28 related elements ripe for similar future academic theoretical and practical study (Fig. 71). Ordering the elements

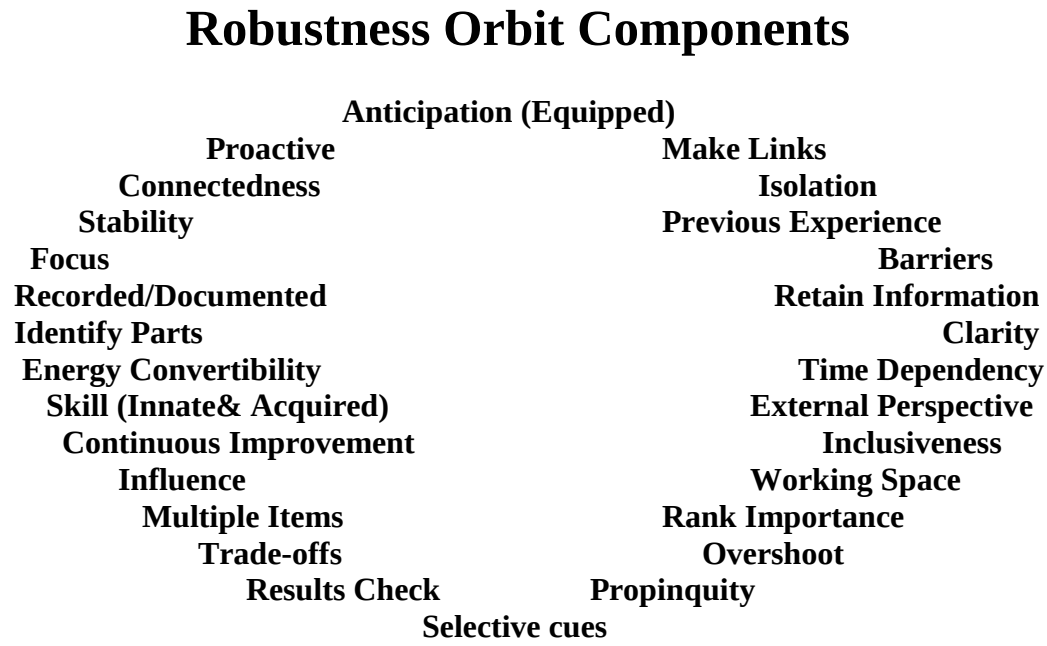


Figure 71. Robustness can be thought to include a collection of elements that work together to provide observed robust behavior, and those elements are shown in an orbit arrangement for ease of recollection, and some priority importance.

at least partially in terms of perceived priority or importance indicated that anticipation rose to the top, and some studies by previous researchers in a theoretical manner supported the necessity to look at anticipation for further study. The progress made with this one important element can be extended to the rest of the elements in the diagram.

Incorporating Anticipation Into Agents

Scale of the situation (in size and time), key connections, operation sequences, tested performance, and design for manufacturability or harvest and processing must all be regarded as key to a framework that converts an abstract formal system into a realistic natural system solution. At all times, the notion of anticipation is still constrained by the Second Law of Thermodynamics, for ongoing increases in entropy will tend to favor and select for the minimalist simplest solution.

Embracing Anticipation In The Future

In all such cases, the robot designer should build in anticipation by realistically considering what simple small set of operations is indeed possible using existing technology, and that which is available in the near future (months to a year). Considering ideas too far removed in space and time are likely to be unsuccessful, and should only be attempted with acknowledgment of the large risks to be undertaken, and indeed the undesirable possible failures. Yet, the past bears many examples of such risks being taken, and we remember those that succeeded, while many times ignoring or forgetting those unsuccessful ones. We remember the Wright brothers' first flights at Kitty Hawk, NC, and the crash of the Hindenburg, yet for quite different reasons. Both had strong impacts on air travel, even for today. One must realize and accept the fact that humans live in a social world that invites robotic agents to serve them, but does not encourage the robots to become more important that the creators themselves. Anticipation must have that benefit of helping humans to have a less menial, more safe, and fulfilling life, while the robotic agents themselves take on the underlying difficulties, and deliver what is expected by human

operators and users. In all cases, the robot must still measure up to human expectations and align with desired anticipation for a future world.

REFERENCES

Ashby, W. (1962). Principles of the self-organizing system. In: Principles of Self-Organization. H. von Foerster and G. W. Zopf, Jr., eds. . New York, Pergamon Press.

Bagodi, V. & Mahanty, B. (2015). Shifting the burden archetype: developing a system dynamics game. *Journal of Modelling in Management*, 10, 380-395.

Boothroyd, G. (1994). Product design for manufacture and assembly. Computer Aided Design, 26, 505-520.

Braitenberg, V. (1986). (originally 1984). Vehicles: Experiments in Synthetic Psychology. The MIT Press, Cambridge, Massachusetts. 152 pp.

Breazeal, C. (2000). Sociable Machines: Expressive Social Exchange Between Humans and Robots, PhD thesis, Massachusetts Institute of Technology.

Brooks, R. (1999). Cambrian Intelligence: the early history of the new AI. MIT Press. Cambridge, Mass. 199 pp

Churchland, P. (1986). Reductive strategies in cognitive neurobiology. Mind, 95, 279-309.

Connell, J. (1990). Minimalist Mobile Robotics: A colony-style architecture for an artificial creature. Academic Press, New York.

Connell, J. (1992). Neural Control of a Mobile Robot. [Coastal Sea Snail and Muramator], Johuco Ltd. Accessed Feb. 4, 2015: http://researcher.watson.ibm.com/researcher/files/us-jconnell/jhc-muram.pdf

Dowling, A., MacDonald, R., & Richardson, G. (1995). Simulation of system archetypes. System Dynamics, 5, 454-463.

Duhigg, C. (2014). The power of habit: Why we do what we do in life and business. Random

House. 383 pp. Reprinted from 2012.

Dupre, J. (1993). The disorder of things: a metaphysical foundations of the disunity of science. Harvard University Press, Cambridge, Massachusetts. 304 pp.

Einstein, A. (1905). Does inertia of a body depend on its energy content? Annals of Physik. In: Einstein's miraculous year: Five papers that changed the face of physics. John Stachel, Ed., 1998. 198 pp. Also accessed Sept. 20, 2015: http://einsteinpapers.press.princeton.edu/vol2-trans/188

Flood, R. (1999). Rethinking the fifth discipline: Learning within the unknowable. Routledge. 213 pp.

Ford, P. (1996). Description of a robot, task, and environment system using the Theory of Affordances. Colorado State University, Masters Thesis.

Goldstein, K. (1995). The Organism: A holistic approach to biology derived from pathological data in man. Zone Books, New York, 422 pp. Reprinted from 1934 & 1963.

Hayward, J. & Boswell, G. (2014). Model Behaviour and the Concept of Loop Impact. System Dynamics Review, 30, 29-57.

Hopper, D. (2008). Comparison of fixed action patterns that are used to direct mobile robot behavior. Colorado State University, Masters Thesis.

Keijzer, F. (2001). Representation and behavior. MIT Press, Cambridge, Massachusetts. 276 pp.

Malcolm, S. & Smithers, T. (1990). Symbol grounding via a hybrid architecture in an autonomous assembly system. *Robotics and Autonomous Systems*, 6, 123-144.

Maslow, A. (1943a). A preface to motivational theory. Psychosomatic Medicine, 5, 85-92.

Maslow, A. (1943b). A theory of human motivation. Psychological Review, 50, 370-396. http://dx.doi.org/10.1037/h0054346

Nadin, M. (2002). Anticipation: The end is where we start from. Lars Muller Publishers. Braden, Switzerland. 127 pp.

Nehmzow, U. (2000). Mobile robotics: a practical introduction. Springer-Verlag: London. 243 pp.

Nehmzow, U. (2006). Scientific methods in mobile robotics: quantitative analysis of agent behaviour. Springer: London, 200 pp.

Pfeifer, R., Iida, F., & Bongard, J. (2005). New Robotics: Design Principles for Intelligent Systems. *Artificial Life*, 11, 99–120. Accessed on November 26, 2013: http://people.csail.mit.edu/iida/papers/ALIFE_pfeiferetal.pdf.

Pfeifer, R. & Bongard, J. (2007). How the body shapes the way we think: A new view of Intelligence. Cambridge, Mass., MIT Press, 394 pp.

Rifkin, J. (1980). Entropy: A new world view. Bantam Books, New York. 302 pp.

Rosen, R. (1991). Life itself: A comprehensive inquiry into the nature, origin, and fabrication of life. Colombia University Press: NY. 285 pp.

Rosen, R. (2012) Anticipatory systems: Philosophical, mathematical, and methodological foundations. 2nd ed. FSR International Series on Systems Science and Engineering , Springer, New York. 472 pp.

Scassellati, B. (2001). Foundations for a Theory of Mind for a humanoid robot. Massachusetts Institute of Technology, Cambridge, MA, PhD Thesis.

Senge, P. (2006). The Fifth Discipline: the art and practice of the learning organization. Doubleday, London, 445 pp. Reprinted from 1990.

Simon, H. (1996). The Sciences of the Artificial. 3rd ed. MIT Press: Massachusetts.

Sterman, J. (2000). Business Dynamics: Systems thinking and modeling for a complex world.

Irwin McGraw-Hill: New York. 982 pp.

Tinbergen, N. (1951). The study of instinct. Oxford University Press, Oxford, England.

Troxell, W. & Troxell, G. (2014). Towards a philosophy of engineering. IEEE Conference on ethics, May 2014.

von Bertalanffy, L. (1968). General systems theory: Foundations, development, applications. George Braziller, New York. 289 pp.

Watson, J. B. (1913). Psychology as the behaviorist views it. Psychological Review, 20, 158-177. See also https://en.wikipedia.org/wiki/John_B._Watson.

Wolstenholme, E. (2003). Towards the definition and use of a core set of archetypal structures in system dynamics. System Dynamics Review, 19: 7–26. doi: 10.1002/sdr.259.

Wolstenholme, E. (2004). Using generic system archetypes to support thinking and modelling. System Dynamics. Review, 20: 341–356. doi: 10.1002/sdr.302

APPENDICES

A1. PERCEPTS

Percepts arise from the area of psychology, as mentioned briefly in the main literature review of Chapter 2, where a percept is the mental recreation of a distal (external) stimulus. For robotics, the mental reference is replaced by an agent. A real world object is the distal stimulus or distal object. Through a physical process (light, sound, etc.) a sensory device/organ is stimulated, in turn using energy to create neural activity (called transduction). The internal raw pattern is the proximal stimulus, which is transmitted to the brain (agent processor) for processing. The resultant recreation of the distal stimulus in the brain is a percept. Overall, perception is creation of mental representations (images or archetypes) using the proximal stimuli derived from distal stimuli. For example, a cat as a distal stimulus is detected by light energy entering the eye to form an image on the retina as a proximal stimulus, and the reconstruction of the image in the brain (agent processor) is the percept (Fig. A.1.1). A bird singing as a distal stimulus uses sound energy to move auditory receptors as the proximal stimulus, and interpretation by the brain (agent processor) is the percept. Intelligent agents choose to act both on individual and sequences of multiple percepts. An agent function maps each percept to an action, and subsequently to the next action. (From: https://en.wikipedia.org/wiki/Perception ; and https://en.wikipedia.org/wiki/Percept_%28artificial_intelligence%29)

For the purposes of this study, a percept is an abstract representation of an element or factor in the niche. Synonyms include: form, rule, habit, image, code, and covenance. In brief, a percept in the FS is the abstraction of an elemental factor in the NS. Anticipation acts by using percepts of

the niche to cue a behavior that is manifest to produce a matching behavior by the agent, and lead to successful observed task achievement. Therefore, the percept is the perceived version of the item in the niche that is used to manifest behavior choice.

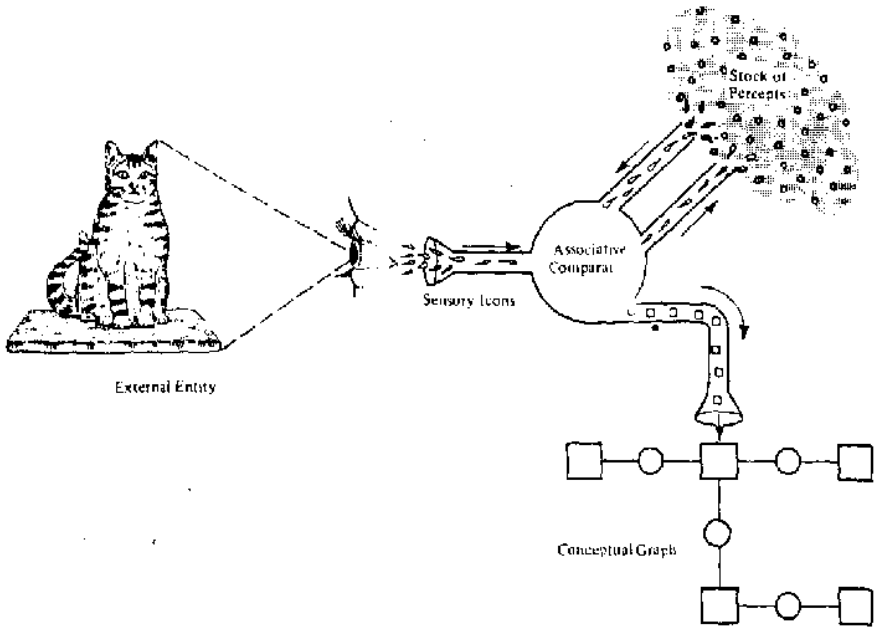


Figure A1.1. Understanding of percepts derived from an external stimulus, transformed to be captured in the brain as abstract conceptual graphs (from: http://kremer.cpsc.ucalgary.ca/courses/CG/L3.html). This representation is quite detailed and complex on the abstraction side (right), and one might contend it resembles Classical Robotics and involved artificial intelligence as opposed to the behavior-based approach that matches action directly to conditions in the niche environment.

A2. BIOPLANT ANALOGY

Plant Development,

Architecture created for the operation of an artificial system such as a robot agent can be applied more generally to a natural biological system for an architecture of the multiple stages for biological plant (bioplant) growth, development, and reproduction (Fig. A2.1). In both the artificial robot system and the natural bioplant system, a niche environment is perceived to form percepts for the current condition. A combination of the percept factors (infrared or IR, light, temperature, etc.) is used to determine a fitness or suitability for a specific behavior to match the niche condition. Within that niche context, a threshold coupling makes a type of selection to manifest the preferred behavior for that condition. For the robot, it is a mathematically integral of the multiple niche percept factors that cue a change in behavior. For the bioplant, the integration occurs in biochemical pathways that create threshold levels of chemical molecules that can cue the initiation, and subsequent continuation, of a change in development stage, or nuances within that stage. Hence, a bioplant changes from the vegetative stage (forming only leaves) to a reproductive stage (forming flower buds) based on a combination of percepts of the niche that cue the change.

Anticipation of future events is tied up with the time constants for response. Both natural bioplants and animals have inherent physical structures and biochemical pathways that lead to preparation for and subsequent manifesting of behavior choices that lead to attaining desired goals for survival and reproduction.

AN in robot systems for AN analogy in bioplant systems.

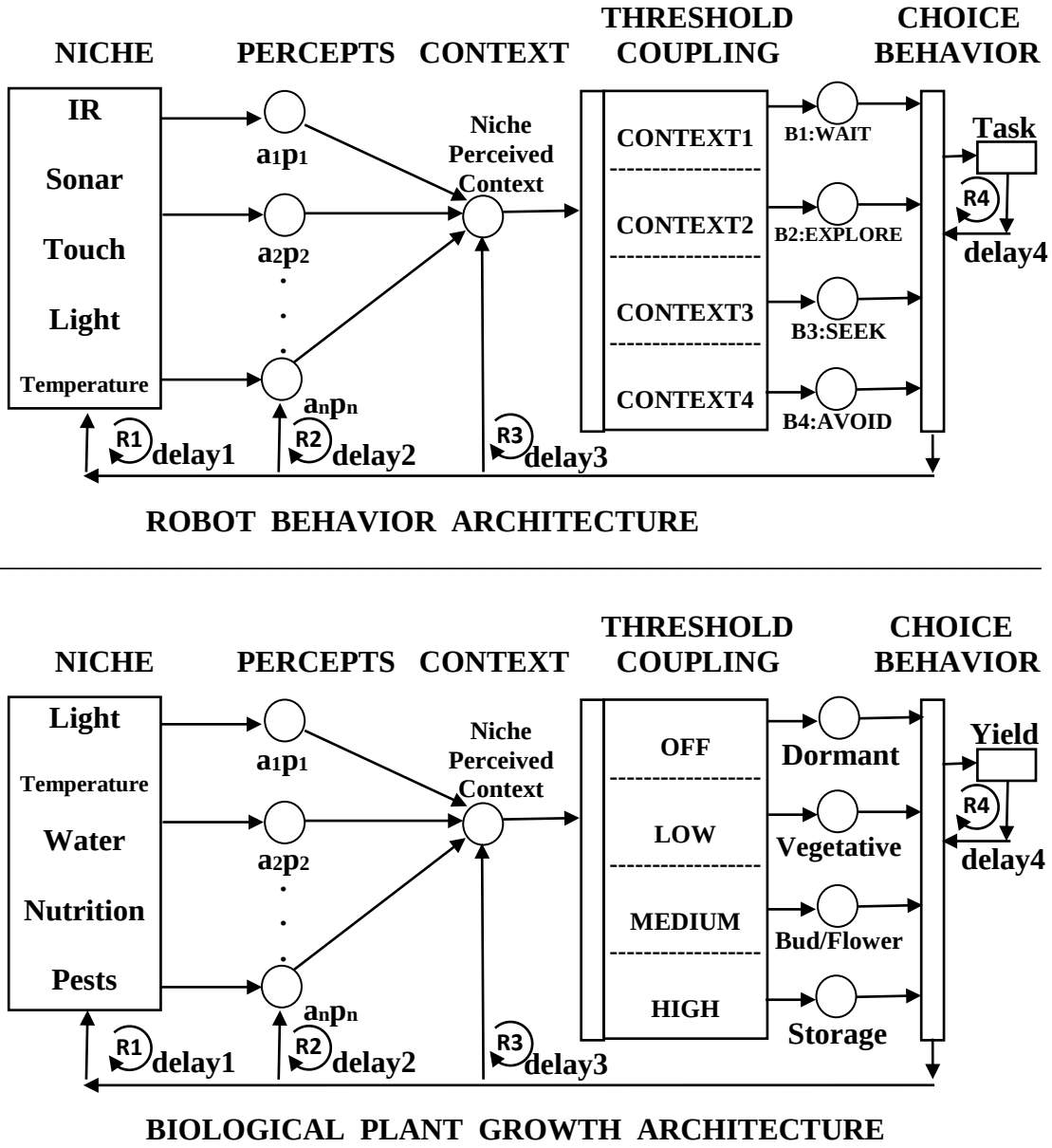


Figure A2.1. Architecture for robotics relating perceived environmental niche to context for robot behavior with reinforcing loop back to the niche (top) and modified for plant architecture (bottom) [causal diagram and flow map].

Evolutionary processes have worked on the ontogeny and phyllogeny of these natural biological living systems (considered open systems in general systems theory) to develop specific behaviors to match a niche, and attain survival outcomes. Anticipation is built into the physical structure

and biochemical pathways that allows behavior change in a shorter time frame than is needed to attain the results that ensure survival. Here anticipation works to benefit the organism, since it acts *before* the outcome is certain, yet the behavior choice leads to a preferred task achievement. The behavior may change again *before* a negative effect is realized if the change in behavior was premature as cued by the niche, and thus not preferred at that time. Thus, the quick and malleable behavior choice is a trait of anticipation that leads to preferred task achievement, and it appears the organism *knew the future* before it actually occurred.

Hops Production Congruence Framework

Hops (*Humulus lupulus*, from Wikipedia) can be modeled within the congruence framework as patterned after Rosen (1991), by considering the elements, 1. Natural System, NS (Hops plant system), 2. Encoding, 3. Formal system model, FS, and 4. Decoding back to the NS.

1. Natural System, NS (Hops plant system): Hops (*Humulus lupulus*, from Wikipedia) are harvested as the unfertilized flower structure that develops on vines grown on upright string structures. The plant is a herbaeceous perennial bearing a single annual crop of fresh hops under natural conditions within the USA that provide adequate light and temperature (daylength sensitivity to induce flowering is not known), while proper nutrition and soil conditions must be provided, along with a mechanical superstructure support system that allows the plants to grow vining in a vertical direction for several meters. Plants are dioecious, meaning separate plants are either male or female, so only the female plants are propagated using asexual means by cuttings, and male plants are removed since pollination actually forms seeds, which is undesirable since it prevents the desired oil production. Various cultivars produce different desired oils that provide specific flavors, and may vary in growth habit as to total production time and total yield. Plants

are subject to known pests such as spider mites and powdery mildew that limit overall yield. Quality of the hops is influenced by freshness of harvest at peak oil production that may be used directly within a few hours in brewing, or the hops may be dried and stored for later use. Brewmasters vary the amount and cultivars of hops used to regulate the traits of beer produced from the brewing and fermentation process. Note there may be considerable difference in a specific biological effect for different crop species. For hops, pollination is a problem (not wanted!), and for strawberries, pollination also is a problem (wanted to be thorough!) which is for the opposite reason!

2. Encoding: Niche conditions must be encoded to capture the specific cues that combine to produce hops plant growth through stages of VEGETATION, BUD INDUCTION, FLOWERING, HOP DEVELOPMENT, HARVEST, and DORMANCY between annual growth cycles. Environmental factors to consider include light (LT), temperature (both for day and night: DT and NT), nutrition (NUT), soil media type (ST), water needs (WT), pest problems (PEST), and mechanical support (MST) needed to hold the crop upright due to the crop vining habit. Temperature is separated into two factors since it is known for plants generally that the rate of plant development is most correlated with night temperature, while plant habit such as stem elongation is controlled by the difference between day and night temperature (DIF). Conditions of light and temperature are presented as total accumulation for each day, while for temperature the values used are an average over the lighted and dark periods. Also for light, if FLOWER or other key response, such as DORMANT, behavior can be determined as cued by some critical daylength, then the length of the lighted and dark periods must be specifically tracked to determine the effect on behavior. Experimentation and measurement of plant traits

must be used to initially abstract the understanding of plant growth into terms that can be described in the mathematical formal system. Possibly overall age (YEARs) the crop has been grown has an effect. Because of the relatively high number of niche factors to consider, the relationships to determine first are those thought to have greatest influence, and the factors above are listed generally in that order. Efforts would be made in the later decoding process to maintain the levels of certain factors constant, in hopes that has the lest significance at some preassigned level. This incorporates the notion of metaphor, for by using conditions that are known to work well in a natural setting, and to reduce pests through environmental means or with pesticides, and to use other relations of nutrition that work well for other related crops, the FS model may include inherent inferences within the model structure, though the true effects of certain factors are never specifically tested and relations determined. Only in extremes or with a change in the cultural growing system, such as to minimize inputs, would the factor relations be determined, and included in subsystems to the original FS model.

3. Formal system model, FS: The formal system includes two rather different types of mathematical relations: 1. ‘Black box’ relations, and 2. Differential equations of physical processes. ‘Black box’ relations can be represented through statistical fit of regression parameters to linear equation terms or certain nonlinear constructs that directly reflect the observed output behavior values in response to specific input values over some range of experimentation and direct measurement of observed output. This abstraction ignores the mechanism in the underlying process, and merely relates output to input through a mathematical construct. At some level of reductionism, this technique is almost always used, and assumes constants for terms that can be readily controlled, and may assume certain factors of no concern

have no effect in the current niche situation. Differential equations are used to capture relations where the mechanism is understood or desired to be represented for further mathematical analysis, such as extrapolation beyond experimental conditions that gave rise to the initial relations, or to study the relationship more detail. Even these attempts to represent the mechanism will eventually reach limits and failure as the desire to simulate extremes become vastly different from the test conditions that are known to work for observed plant system growth. This principle follows that of Simon (1996), where he contends only local levels of input in space and time are of concern, and the farther one moves from known conditions, the less sure one is of the results to expect. In this way, anticipation of results is limited to conditions that are expected and known, while widely varying conditions would actually be expected to produce results that differ greatly from those observed for known conditions. The two types of equations can be combined to capture the entailments of inference that will make a behavior choice (a particular output value) based on multiple input cues. The black box regression equations may be thought of as:

$$BIOMASS = f(LT, DT, NT, NUT, PEST, WT, YEAR), \quad (A1.1)$$

where each of these factors was define above.

Generally, these may be linear, quadratic or cubic regressions for each of the pertinent factors, or may involve a sinusoidal varying input of any of the factors.

Slightly differently, the differential equations may be represented as:

$$d(BIOMASS)/dt = f(An, An', An'') \quad (A1.2)$$

where An, and the successive primes are the derivatives with respect to time. Either of the two types of equations might be used for a trait (e.g., height, time to harvest) or the two types might be used in unison to capture nuances of the relations in the niche and in time. Simulations may

be run for various inputs of interest, such as reducing inputs to lowest possible levels while maintaining desired quality or timing for harvest. Results of the simulations point to the direction of interest for applying the results found in the FS to improve results in the NS.

4. Decoding back to the NS: Once simulation has determined courses of action or niche conditions of interest to be used in the NS, the simulated results must be decoded back to the congruent entailments of causation in the NS. On the simplest level, decoding for a greenhouse of growth chamber allows setting of fairly precise niche conditions, and thus the resulting behavior can be observed to be congruent or not with simulated results. For a more natural setting, measured environmental niche conditions must be input to the model, and the results compared with observations to affirm congruence. If results are not as congruent as desired, several items should be considered for making changes to the FS model, including scaling in space and time, key operations, forming connections, sequential ordering, comparison of options, and design for manufacturing and assembly type operations that include harvest. Each of these can be described for the hops production with a couple examples that follow.

Scaling aspects (Large-scale):

Desired: An arena must be large enough to produce a quantity of hops for commercial use.

Decode: A crop size must include enough plants so minimum produced can make a batch.

Scaling aspects (Small-scale):

Desired: Individual flowers should be close together for easy harvest.

Decode: Niche conditions should limit growth rate to keep flowers close together.

A3. ANTICIPATION SIMULATION PROGRAM CODE

ANSIM CODE

Program for ANSIM (file: ANSimDelay20160121R.slx; ver. 10-28-2016): .

NICHE: Subsystem NICHE LAYOUT

```
function [xgrid, ygrid, WallObj] = NicheLayoutfcn(ctimein1)
%#codegen
global NicheLayout;%from NICHE section ML function; 201X201X1 init
%NicheLayout(x,y,zval)= (x,y )biomass/locations; NicheLayout(xlayout,ylayout)
%NicheLayout(:,:,1)= xval, min to max
%NicheLayout(:,:,2)= yval, min to max
%NicheLayout(:,:,3)= zval wall or object: 0=open, 1=solid
%center (0,0,:) at NicheLayout(101,101,:)

coder.extrinsic('sprintf', 'strcat');
coder.extrinsic('format','display');

if (ctimein1 == 0);%create Niche at time=0
%Decide not to use negative numbers for range covered. 2015-12-5Sa
%Since 28cm=11in is real bot, using 200 as max range

%[m,n] = size(obj) from
http://www.mathworks.com/help/distcomp/size.html?requestedDomain=www.mathwork
s.com
[xmaxNL,ymaxNL, zmaxNL] = size(NicheLayout);

%......Wall #1 bottom
xmin=0;%-10;
ymin=0;%-10;
xminwall01=0;
xmaxwall01=xmaxNL;%80;%5;
yminwall01=0;%-0.1;
ymaxwall01=1.0;%0.0;

%...... wall #2 top
xminwall02=0;
xmaxwall02=xmaxNL;%50;%5;
%subtract 3 & 2 below to make in 99 and 100 locations,
%  since matrix contents is: index-2
%  specifically: NicheLayout(xlayout,ylayout,1)= xmin + (xlayout-2);
yminwall02=ymaxNL-3;%49;%-0.1;
ymaxwall02=ymaxNL-2;%50.0;%0.0;
%......

%......wall #3 left
xminwall03=0;
xmaxwall03=1;
%subtract 3 & 2 below to make in 99 and 100 locations,
%  since matrix contents is: index-2
%  specifically: NicheLayout(xlayout,ylayout,1)= xmin + (xlayout-2);
yminwall03=0;%49;%-0.1;
ymaxwall03=ymaxNL-2;%50.0;%0.0;
```

```
%......

%......wall #4 right
xminwall04=xmaxNL-3;
xmaxwall04=xmaxNL-2;
%subtract 3 & 2 below to make in 99 and 100 locations,
%  since matrix contents is: index-2
%  specifically: NicheLayout(xlayout,ylayout,1)= xmin + (xlayout-2);
yminwall04=0;
ymaxwall04=ymaxNL-2;
%......

%......objects #1 to # 2
% 70 73 73  70 70
xminobj01=70; xmaxobj01=73; yminobj01=70; ymaxobj01=73;%upper right
%xminobj01=50; xmaxobj01=53; yminobj01=50; ymaxobj01=53; %center
%xminobj01=51; xmaxobj01=54; yminobj01=21; ymaxobj01=24;%original

xminobj02=17; xmaxobj02=20; yminobj02=20; ymaxobj02=23;%lower left
%xminobj02=21; xmaxobj02=24; yminobj02=51; ymaxobj02=54;%original
%......


xyNiche=' ';
spcolon=':';

for ylayout= ymaxNL : -1 : 1;%52;%total 52 elements; lowest as -1: see eqs
below
%for ylayout= 52 : -1 : 1;%52;%total 52 elements; lowest as -1: see eqs below
%index must be 1 or greater (no index zero)%NicheLayout(xlayout,ylayout,2)=
ymin + (ylayout-1);%((ylayout-1)/10.0) ;
%for ylayout= 201 : -1 : 1;%1 : 121;%201;%total 201 elements; 101 is middle=0
%NicheLayout as global variable on highest level system window
%zeros(52,52,3)%previous took 10 min to run
%zeros(201,201,3)%previous took 10 min to run

%NicheLayout initialized as a global: NicheLayout(201,201,3)
%if not start  at either end, matrix will have unwanted zeros
%if not finish at either end, matrix will have unwanted zeros

 yrowprt= ': ';
 for xlayout= 1 : xmaxNL;%52;%total 52 elements; lowest as -1: see eqs below
 %for xlayout= 1 : 52;%52;%total 52 elements; lowest as -1: see eqs below
 %index must be 1 or greater (no index zero)
 %NicheLayout(xlayout,ylayout,1)= xmin + (xlayout-1);% ((xlayout-1)/10.0 ) ;
 %for xlayout= 1 : 201;%201;%total 201 elements; 101 is middle=0
  NicheLayout(xlayout,ylayout,1)= xmin + (xlayout-2);% ((xlayout-1)/10.0 ) ;
  NicheLayout(xlayout,ylayout,2)= ymin + (ylayout-2);%((ylayout-1)/10.0) ;
  %NicheLayout(xlayout,ylayout,1)= xmin + (xlayout-1);% ((xlayout-1)/10.0 ) ;
  %NicheLayout(xlayout,ylayout,2)= ymin + (ylayout-1);%((ylayout-1)/10.0) ;

  if (ylayout == 5) && (xlayout == 5);
    spxlayout= xlayout;%inner loop: xlayout
    %spylayout= xlayout;
  end;%if (ylayout== 5)
```

```
  %=====add wall and objects here


  %=======create a wall #1 Bottom of Graph===============
  x1=xlayout;
  y1=ylayout;
  if (NicheLayout(x1,y1,1) >= xminwall01) && (NicheLayout(x1,y1,1) <=
xmaxwall01);
    if (NicheLayout(x1,y1,2) >= yminwall01) && (NicheLayout(x1,y1,2) <=
ymaxwall01);
         NicheLayout(x1,y1,3)= 1;
    end;%if (NicheLayout(x1,y1,2) >= ...
  end;%if (NicheLayout(x1,y1,1) >= ...
  %=======end create a wall #1 Bottom of Graph===============

  %=======create a wall #2 Top of graph===============
  %x1=xlayout;
  %y1=ylayout;
  if (NicheLayout(x1,y1,1) >= xminwall02) && (NicheLayout(x1,y1,1) <=
xmaxwall02);
    if (NicheLayout(x1,y1,2) >= yminwall02) && (NicheLayout(x1,y1,2) <=
ymaxwall02);
         NicheLayout(x1,y1,3)= 1;
    end;%if (NicheLayout(x1,y1,2) >= ...
  end;%if (NicheLayout(x1,y1,1) >= ...
  %=======end create a wall #2 Top of Graph===============

  %=======create a wall #3 Left of graph===============
  %x1=xlayout;
  %y1=ylayout;
  if (NicheLayout(x1,y1,1) >= xminwall03) && (NicheLayout(x1,y1,1) <=
xmaxwall03);
    if (NicheLayout(x1,y1,2) >= yminwall03) && (NicheLayout(x1,y1,2) <=
ymaxwall03);
         NicheLayout(x1,y1,3)= 1;
    end;%if (NicheLayout(x1,y1,2) >= ...
  end;%if (NicheLayout(x1,y1,1) >= ...
  %=======end create a wall #3 Left of Graph===============

  %=======create a wall #4 Right of graph===============
  %x1=xlayout;
  %y1=ylayout;
  if (NicheLayout(x1,y1,1) >= xminwall04) && (NicheLayout(x1,y1,1) <=
xmaxwall04);
    if (NicheLayout(x1,y1,2) >= yminwall04) && (NicheLayout(x1,y1,2) <=
ymaxwall04);
         NicheLayout(x1,y1,3)= 1;
    end;%if (NicheLayout(x1,y1,2) >= ...
  end;%if (NicheLayout(x1,y1,1) >= ...
  %=======end create a wall #4 Right of Graph===============

% {
%add space after % to reactivate block
%start of block comment for objects 2 & 1
  %=======create an object #1 ===============
```

```
  if (NicheLayout(x1,y1,1) >= xminobj01) && (NicheLayout(x1,y1,1) <=
xmaxobj01);
    if (NicheLayout(x1,y1,2) >= yminobj01) && (NicheLayout(x1,y1,2) <=
ymaxobj01);
         NicheLayout(x1,y1,3)= 1;
    end;%if (NicheLayout(x1,y1,2) >= ...
  end;%if (NicheLayout(x1,y1,1) >= ...
  %=======end create an object #1 ===============

% }
%add space after % to reactivate block
%end of block comment for object 1

% {
%add space after % to reactivate block
%start of block comment for objects 2
  %=======create an object #2 ===============
  if (NicheLayout(x1,y1,1) >= xminobj02) && (NicheLayout(x1,y1,1) <=
xmaxobj02);
    if (NicheLayout(x1,y1,2) >= yminobj02) && (NicheLayout(x1,y1,2) <=
ymaxobj02);
         NicheLayout(x1,y1,3)= 1;
    end;%if (NicheLayout(x1,y1,2) >= ...
  end;%if (NicheLayout(x1,y1,1) >= ...
  %=======end create an object #2 ===============
% }
%add space after % to reactivate block
%end of block comment for objects 2 & 1



 end;%for xlayout= 1 : 201;%201;


 %coder.extrinsic('format','display');
 spy1= ylayout;
 %spx1= xlayout;

  if (ylayout >= -1)%all rows
  %if (ylayout >= (yminwall01-1)) && (ylayout <= (ymaxwall01+1))%rows having
ones in them
  %if (ylayout >= 90) && (ylayout <= 110)%rows having ones in them
   spy1= ylayout;
   %spx1= xlayout;
   spNR= NicheLayout(:,ylayout,3);
   ID= sprintf('%03d', ylayout);
   ID2= strcat (ID, char(58) );%add a colon: as char(58)
   %http://www.mathworks.com/help/matlab/ref/char.html?searchHighlight=char
   %NH= sprintf('%0.0f', xlayout, spNR )
   %NH2= strcat (ID, sprintf('%0.0f', spNR ) )
   NH3= strcat (ID2, sprintf('%0.0f', spNR ) );%concatenate only 2 strings at
a time
   disp(NH3) %displays without variable name
   %sptNR= sprintf('%0.0f', NicheRow )
   %sptNR= sprintf('%0.0f', trNicheRow )
```

```
   %yrowprt= strcat (yrowprt , NicheRow1 )

   % %spfc1= sprintf('%0.0f', tANiche(:,1));
   % %names = strcat(spfc3, {':::'}, spfc2);

   %xyNiche = [ xyNiche char(10) yrowprt]
   %names3 = [ names3 char(10)  spfc4]


  end%if(x1 <= 5)


 %elseif (ylayout <= 120) %syntax may not be correct here
   %return %exits this call function: stop niche definition
 end%for ylayout= 1 : 121;%201;


 %---add routine here to take rows of NicheLayout(:,:,3)
 %need to correct the code below

 %for 201:-1:1
   %print x row values successively
   %alternate is to generate nested loop with y on outer loop, x as inner
loop
 %end%for 201:-1:1

 %---test an array output
  %...example
  % Creating Multi-Dimensional Arrays
  % Multidimensional arrays in MATLAB are created the same way as
  % two-dimensional arrays. For example, first define the 3 by 3 matrix, and
  % then add a third dimension.

  %A = [5 7 8;
  %     0 1 9;
  %     4 3 6];
  %A(:,:,2) = [1 0 4;
  %            3 5 6;
  %            9 8 7];
  %...end example

  %
  %Format a floating-point number using %e, %f, and %g specifiers.
  %A = 1/eps;
  %str_e = sprintf('%0.5e',A)
  %str_f = sprintf('%0.5f',A)
  %str_g = sprintf('%0.5g',A)


 %---end test array output

%====end of add wall and objects here

%============Add surface plot of NicheLayout (once after generated)
```

```
%http://www.mathworks.com/help/matlab/ref/surface.html?searchHighlight=surfac
e
%surface(X,Y,Z,C) plots the parametric surface specified by X, Y, and Z, with
color specified by C.
%http://www.mathworks.com/help/matlab/ref/primitivesurface-properties.html
%CData is of type uint8, then [0 0 0] corresponds to black and [255 255 255]
corresponds to white.

%hold on; %keeps previous plot and adds new lines to it.
%hold off; %replaces previous plot.
hold off;%not seem to work between runs
hNL= figure('Name','Niche Layout','NumberTitle','off',... %figure with handle
hNL
            'MenuBar', 'none', 'ToolBar', 'none');%figure with handle hNL
            %'Visible', 'on' );%figure with handle hNL
%255 is for some othether system
%ClrData= ( [255 255 255] ; [0 0 0] );%syntax not correct
%map = [255, 255, 255
%       0, 0, 0];
%colormap([255  255  255; 0  0  0]);%black=255 255 255; white= 0 0 0

%http://www.mathworks.com/help/matlab/ref/colorspec.html
%colormap([1  1  0; 0  1  1])
%colormap([1 1 1; 0 0 0]);%white= 1 1 1; black=0 0 0;

%blending of colors: no grid...
%http://www.mathworks.com/help/matlab/visualize/representing-a-matrix-as-a-
surface.html
%http://www.mathworks.com/help/matlab/ref/surface.html?searchHighlight=surfac
e
%surface(XD,YD,ZD,C,...
%    'FaceColor','texturemap',...
%    'EdgeColor','none',...
%    'CDataMapping','direct')

% to remove grid, so use: 'EdgeColor','none',...

%CData is color data, as zero or one for this matrix
hNHs1=surf( NicheLayout(:,:,1) , NicheLayout(:,:,2) , NicheLayout(:,:,3),...
    'CData',  NicheLayout(:,:,3),...
    'EdgeColor','none');%no grid drawn? or set to 'w' for white or [1 1 1]
colormap([1 1 1; 0 0 0]);%white= 1 1 1; black=0 0 0;
%surf differs from surface
%hNHs1=surface(NicheLayout(:,:,1) , NicheLayout(:,:,2) ,
NicheLayout(:,:,3),...
%         'CData',  NicheLayout(:,:,3) );

%http://www.mathworks.com/help/matlab/ref/axis.html
%axis(limits) sets the limits for the current axes.
%If the current axes is a Cartesian axes, then specify limits as a
%four-element vector of the form [xmin xmax ymin ymax] to set the
%x-axis and y-axis limits. To also set the z-axis limits,
%specify a six-element vector. To also set the color limits, specify an
%eight-element vector. If the current axes is a polar axes, then specify
%limits as a four-element vector to set the theta-axis and r-axis limits.
%axis([xmin xmax ymin ymax zmin zmax] );
```

```
%axis([-10  110  -10  110  0    2   ] );
 axis([ -5  105   -5  105  0    2   ] );

%hold on; %keeps previous plot and adds new lines to it.
%View the object from directly overhead: azimuth, elevation.
%az = 0; %el = 90;
view(0, 90);%azimuth, elevation
%view(-35,45);
%refresh(hNL);%not needed?
%refresh(hNHs1);%block error: why%

spctimein1= ctimein1 %print to see if reach here

%Ctrl-[break/pause] key to stop the program
%next 3 lines force an error and stops program run 'Name' not allowed here.
%hNHs2=surface(NicheLayout(:,:,1) , NicheLayout(:,:,2) ,
NicheLayout(:,:,3),...
%        'Name','JunkNiche LayoutJunk',...
%         'CData',  NicheLayout(:,:,3) );

%==========end Add surface plot

end;%if (ctimein1 ==0);%create Niche at time=0

%y = NicheLayout;
xgrid = NicheLayout(:,:,1);
ygrid = NicheLayout(:,:,2);
WallObj  = NicheLayout(:,:,3);
```

PERCEPTS: Subsystem ML IRSen01 value

```
function [IRSen01, yGain, yIR, xOut, yOut, aGainOut, bGainOut, BotHead,
IRHead,xFixMin, yFixMin, xFixMax, yFixMax]= IRSen01fcn(u)
%function [IRSen01, BotHead, IRHead]= IRSen01fcn(u)%#codegen
%Use to determine if IRSen01 detects a wall.
global NicheLayout;%from NICHE section ML function; 201X201X1 init
%know previous behavior for changes
global aGain;
global bGain;%keeps values between loops
global daGain;
global dbGain;%keeps values between loops
global heading;
global speed;%keeps values between loops
global HoldTimeAll dtimeall ;%keeps values between loops
global IRSen01Dist IRSen01Min IRSen01Max;%Use in AHEAD_AVOID and/or
FIND_SEEK;
global AN01;%AN 0=off, 1=0n
global xReadingFixPlot yReadingFixPlot;%Plot IR distance boundary
global xReadingMinPlot yReadingMinPlot;%Plot Min IR distance boundary

persistent IROffset IRHeading IR01Prev;%headng IR sensor is pointing
persistent IRReading;%distance reading convert from voltage
persistent xReading yReading;%
persistent xReadingMin yReadingMin;%
persistent IRDistMax;%closeness value to trigger sensor (may use later?)
%persistent HoldyTest;%allow only +1 yTest increment (previous value)

coder.extrinsic('sprintf', 'strcat');
coder.extrinsic('format','display');
coder.extrinsic('find');

if isempty(IROffset);
 IROffset= pi()/6;% pi()/6= 180/6= 30 degrees% pi()/4= 180/4= 45 degrees
 %for ANNum calc in EXPLORE as 30 default; need reset for any other angle
 % actually in EnabledSybsystem/ML EXPLORE C1*
 IRReading= 100;%init as far from any object
 xReading=0;
 yReading=0;
 xReadingMin=0;
 yReadingMin=0;
 xTest=0;
 yTest=0;
 IRDistMax= IRSen01Max;%60;%28;%initial value %Setting as 28cm=11in.
closeness boundary to cue AVOID
 %2016-7-22F global variables handle in multiple routines.
 %Must also change in Context function:  Fitnessfcn(ctimein02,u)
 %about line 89: if (IR01cm < 40);%28);%//28cm=11in//25 cm=10in
  %may set IRDistMax in another function
     %Also change in AHEAD_AVOID function line 57 as base to divide into.

 %HoldyTest=1;%allow only +1 yTest increment (previous value)
 %NewyTest=1;%allow only +1 yTest increment (new calced value)
end;%if isempty(IROffset)

IRHeading= heading - IROffset;%heading IR sensor is pointing; neg=CW
```

```
%initialize
IR01=100;
yTest=1;
HoldyTest=1;%allow only +1 yTest increment (previous value)
NewyTest=1;%allow only +1 yTest increment (new calced value)



%===see items needed below: trig
%also: AVOID change heading?


%%%%%%%%%%%%%%%%%%%  ANticipation additions %%%%%%%%%%%%%%%%%%%%
 %set test value for distance with or w/o AN 2016-7-25; 4:30pm.
 if (AN01 > 0);%AN is on
  IRSen01Test= IRSen01Max;%wide range
 else;%AN01=0: AN off
  IRSen01Test= IRSen01Min;%narrow range
 end;%if (AN01 > 0);%AN is on
%%%%%%%%%%%%%%%%%  END ANticipation additions %%%%%%%%%%%%%%%%%%%%


%(aGain,bGain) is (x,y) in area; assumed as IRSen01 location (stretch?)
%(daGain,dbGain) are previous moves in (dx, dy) directions.
%Agent is pointing in heading absolute direction in area.
%IRSen01 is pointing in IRHeading absolute direction in area.

%2016-09-23F revised test distance
xDelta= IRSen01Test* cos (IRHeading);
yDelta= IRSen01Test* sin (IRHeading);
xReading= aGain + IRSen01Test* cos (IRHeading);
yReading= bGain + IRSen01Test* sin (IRHeading);
xReadingMin= aGain + IRSen01Min* cos (IRHeading);
yReadingMin= bGain + IRSen01Min* sin (IRHeading);

%Old calc using IRDistMax
%xDelta= IRDistMax* cos (IRHeading);
%yDelta= IRDistMax* sin (IRHeading);
%xReading= aGain + IRDistMax* cos (IRHeading);
%yReading= bGain + IRDistMax* sin (IRHeading);

%next line for debug to stop at a specific time range.
if (HoldTimeAll > 3.4 && HoldTimeAll < 3.6);
%if (HoldTimeAll > 3 && HoldTimeAll < 4);
   ARunTime2=HoldTimeAll;%exit for loop
   Junk=1;
end;%(HoldTimeAll > 3 && HoldTimeAll < 4);

%++++++++++++++++++++++++++++++++
%must check if pointed directly +/-x or +/-y for infinite value
%================================

%=====is IRHeading in positive x dir
if (xReading > aGain);%positive x dir
```

```
  xStepOne= 1;%add in extra step
elseif (xReading == aGain);%along y axis ( need double equal signs)
  xStepOne= 1;
else;%xReading < aGain: neg x dir
  xStepOne=-1;
end;%if (xReading > aGain);%positive x dir
%============================

%=====is IRHeading in positive y dir
if (yReading > bGain);%positive y dir
  yStepOne= 1;%add in extra step
elseif (yReading == bGain);%along x axis (need double equal signs)
  yStepOne= 1;
else;%xReading < aGain: neg x dir
  yStepOne=-1;
end;%if (xReading > aGain);%positive x dir
%============================

[mMax,nMax,qMax ] = size(NicheLayout);
   %[m,n] = size(X) example f size return ML help

xFixMin=min(fix(aGain), fix(xReading) );
xFixMax=max(fix(aGain), fix(xReading) );
yFixMin=min(fix(bGain), fix(yReading) );
yFixMax=max(fix(bGain), fix(yReading) );

[xMinIndex, xcol1]= find(NicheLayout(:,:,1)>=xFixMin,1,'first');
  %k = find(X,4,'last')
[yMinIndex, ycol1]= find(NicheLayout(:,:,2)>=yFixMin,1,'first');

%==========IR range aGain to xReading, bGain to yReading=======
aGainFix=max(fix(aGain), 1 );
bGainFix=max(fix(bGain), 1 );
%Plot of xReading & yReading
xReadingFixPlot=fix(xReading);%Plot on final path graph
yReadingFixPlot=fix(yReading);%Plot on final path graph

xReadingMinPlot=fix(xReadingMin);%Plot on final path graph
yReadingMinPlot=fix(yReadingMin);%Plot on final path graph



%xReading and yReading can be calculated outside arena
xReadingFix=max(fix(xReading), 1 );
xReadingFix=min(fix(xReading), mMax);%100 );%above [mMax,nMax,qMax ] =
size(NicheLayout);
yReadingFix=max(fix(yReading), 1 );
yReadingFix=min(fix(yReading), nMax);%100 );% above[mMax,nMax,qMax ] =
size(NicheLayout);
%Bounded Plot of xReading & yReading
%xReadingFixPlot=min(100,xReadingFix);%Plot on final path graph
%yReadingFixPlot=min(100,yReadingFix);%Plot on final path graph
%xReadingFixPlot=max(0,xReadingFixPlot);%Plot on final path graph
%yReadingFixPlot=max(0,yReadingFixPlot);%Plot on final path graph
%==========END IR range aGain to xReading, bGain to yReading=======
```

```
%..............................
if (xFixMin <  NicheLayout(1,1,1));
   spNR= aGain;
   ID= sprintf('%03d', xFixMin);
   ID2= strcat (ID, char(58) );%add a colon: as char(58)
   %http://www.mathworks.com/help/matlab/ref/char.html?searchHighlight=char
   %NH= sprintf('%0.0f', xlayout, spNR )
   %NH2= strcat (ID, sprintf('%0.0f', spNR ) )
   %formatSpec = 'The array is %dx%d.';
   %A1 = 2;
   %A2 = 3;
   %str = sprintf(formatSpec,A1,A2)
   NH3= strcat (ID2, sprintf(' =xFixMin, HALTED: aGain= %0.0f', spNR )
);%concatenate only 2 strings at a time
   disp(NH3) %displays without variable name
elseif (xFixMax > NicheLayout(mMax,nMax,1) );
   spNR= aGain;
   ID= sprintf('%03d', xFixMax);
   ID2= strcat (ID, char(58) );%add a colon: as char(58)
   %http://www.mathworks.com/help/matlab/ref/char.html?searchHighlight=char
   %NH= sprintf('%0.0f', xlayout, spNR )
   %NH2= strcat (ID, sprintf('%0.0f', spNR ) )
   %formatSpec = 'The array is %dx%d.';
   %A1 = 2;
   %A2 = 3;
   %str = sprintf(formatSpec,A1,A2)
   NH3= strcat (ID2, sprintf(' =xFixMax, HALTED: aGain= %0.0f', spNR )
);%concatenate only 2 strings at a time
   disp(NH3) %displays without variable name
else
 %continue
end;%(xFixMin <  NicheLayout(1,1,1));
%..............................



%need to test this in its own little program...
%fix rounds to the nearest integer toward zero (up if neg).
xfor=0;%flag leave exit for loop
yfor=0;%flag leave exit for loop
IR01Prev=IR01;

%*********************************************************************
%2016-1-22F; 7pm
%need to locate max and min values in NicheLayout(x,y,z) to stay in niche
%*********************************************************************

%LoopMin=min(aGain, xReading);
%LoopMax=max(aGain, xReading);
%*********************************************************************

%******next line wrong: has upper end as neg, so loops once.************
%for xTest= fix(aGain)+ xStepOne : 1 : fix(xReading)+ xStepOne;%xStepOne set
above based on pos or neg direction
```

```
%**********Need work here. 2016-1-25M, 2:34pm
%see changed (but seems wrong) code in : IRSen01fcnSeemsOverWriting.m
%next 2 lines need revision for looping of x and y distances
for xTest= aGainFix : xStepOne : xReadingFix;%xStepOne: aGain to xReading
%for xTest= xFixMin : 1 : xFixMax;%xStepOne may not be needed anymore
%for xTest= fix(aGain)+ xStepOne : xStepOne : fix(xReading)+
xStepOne;%xStepOne set above based on pos or neg direction
 %floor rounds towards neg infinity: rounds down, or truncate.
 %first time through loop is tested below as:  if (xTest<= xFixMin);%first
time through loop
 %------- for bGainFix to yReadingFix
 if (aGainFix ~= xReadingFix);%not divide by zero
  xFactor=abs(  (xTest-aGainFix)/(fix(xReading)-aGainFix)   );
  yTest= bGainFix + floor (xFactor * (fix(yReading)-bGainFix) );%linear y
value along line
  %Chnaged two lines below to get range for XReading & YReading correct.
  % 2016-7-8F; 2:30pm
  %xFactor=abs(  (xTest-aGainFix)/(xReadingFix-aGainFix)   );
  %yTest= bGainFix + floor (xFactor * (yReadingFix-bGainFix) );%linear y
value along line
  testXReading=fix(xReading);%view variable values
  testYReading=fix(yReading);%view variable values
  testYReading;%view variable values
 else;
  yTest = bGainFix;%keep at one value: maybe should be changed...step through
y values
 end;% if (xFixMax > xFixMin)
 %-------END for bGainFix to yReadingFix



 %-------replaced for  bGainFix to yReadingFix
 %if (xFixMax > xFixMin);%not divide by zero
 % yTest= yFixMin + floor ( ( (xTest-xFixMin)/(xFixMax-xFixMin))   *
(yFixMax-yFixMin) );%linear y value along line
 %else;
 % yTest = yFixMin;%keep at one value: maybe should be changed...step through
y values
 %end;% if (xFixMax > xFixMin)
 %-------END replace for  bGainFix to yReadingFix

 %for yTest= yFixMin : 1 : yFixMax;%yStepOne may not be needed anymore


  %Trig still needed even if min and max values used for both x and y
directions
    %want a line traveled, not a square
  %yTest= fix(bGain) + fix( (xTest-aGain)*sin(IRHeading) );%yStepOne set
above based on pos or neg direction

  %----for range: bGain to yReading; may not be correct place------
  if (xTest== aGainFix);%first time through loop
   HoldyTest=  yTest;%force OK for 1st yTest value
   xReading;
   yReading;
```

```
   IRHeading;
   HoldTimeAll;
   HoldyTest=max(yTest, 1);
  end;%if (xTest== xFixMin);
%----not seem to be correct place------


 %----before range bGain to yReading; may not be correct place------
 % if (xTest<= xFixMin);%first time through loop
 %  HoldyTest=  yTest;%force OK for 1st yTest value
 %  xReading;
 %  yReading;
 %  IRHeading;
 %  HoldTimeAll;
 %  HoldyTest=max(yTest, 1);
 % end;%if (xTest== xFixMin);
%----not seem to be correct place------

  %constrain in NichELayout index bounds
  xTest= max(xTest, 1);% NicheLayout(1,1,1) );% 1 is y value
  xTest= min(xTest, mMax);%NicheLayout(mMax,nMax,1) );%1 is y value
  yTest= max(yTest, 1);%NicheLayout(1,1,2) );% 2 is y value
  yTest= min(yTest, nMax);%NicheLayout(mMax,nMax,2) );%2 is y value
  %.....is the NicheLayout value a one (solid)


  %........time check.................................
  if (HoldTimeAll<10);
      ATime= HoldTimeAll;
  end;%if (HoldTimeAll<10);
  %.........end of  time check......................

  %look for wall location-----------------------------------
  %if ((yTest >= 2) && (yTest <= 4));%wall in y dir
  %  if ((xTest >= 1) && (xTest <= 50));%may go to 80; wall in x dir
  %    xTextck= xTest;
  %    yTestck= yTest;
  %    Nicheck= NicheLayout(xTest,yTest,3);
  %    StopHereck=0;
  % end;%if (xTest >= 2) && (xTest <= 4);%wall in x dir
  %end;%if (yTest >= 2) && (yTest <= 4);%wall in y dir
  %Endlook for walllocation-----------------------------------

  NewyTest= yTest;%allow only +1 or -1 increments in yTest

  for incTest= HoldyTest : yStepOne : NewyTest;%increment only by 1 for each
test
  %for incTest= HoldyTest : 1 : NewyTest;%increment only by 1 for each test
   if (xTest< 1 );
    xTest;
    xTest=1;
    %break;%exit for loop for yTest
   end;%if (xfor == 1)

   if (incTest < 1);
```

```
    incTest;
    incTest=1;
    %break;%exit for loop for yTest
   end;%if (xfor == 1)

   if ( NicheLayout(xTest,incTest,3)==1 );%solid object detected at this
distance
     xfor=1;
     yfor=1;
     aRunTime1=HoldTimeAll;%previous time likely...
     xdist= NicheLayout(xTest,incTest,1)-aGain;%xTest-aGain;%cos(xTest-
aGain);
     ydist= NicheLayout(xTest,incTest,2)-bGain;%yTest-bGain;%sin(yTest-
bGain);
     IR01= sqrt(xdist*xdist + ydist*ydist);%Pythagorus distance
     IR01=min(IR01Prev,IR01);
     IR01Prev=IR01;
   elseif (yfor == 0);%not solid object and none yet found in IR path scan
     IR01=100;%set outside of range to cue close to object
   else;
    %continue
   end;%if ( NicheLayout(xTest,yTest,3)==1 )

   if (yfor == 1);
      break;%exit for loop for yTest
   end;%if (xfor == 1)

  end;%for incTest= HoldyTest : 1 : NewyTest;
  %...................

 %In wrong place?: HoldyTest= yTest;%hold for test increment=1 for each test.
 %end;%for yTest= yFixMin : 1 : yFixMax;%yStepOne may not be needed anymore

 HoldyTest= yTest;%hold for test increment=1 for each test.

 if (xfor == 1);
  break;%exit for loop for xTest
 end;%if (xfor == 1)

  %xOut=fix(xTest);%
  %yOut=fix(yTest);%
end;%for xTest= xFixMin : 1 : xFixMax;%xStepOne may not be needed
anymore%was:for xTest= aGain : 1 : fix(xReading)...

%next line for debug to stop at a specific time range.
if (HoldTimeAll > 6.9 && HoldTimeAll < 7.1);
%if (HoldTimeAll > 3 && HoldTimeAll < 4);
   ARunTime2=HoldTimeAll;%exit for loop
   Junk=1;
end;%(HoldTimeAll > 3 && HoldTimeAll < 4);

%BotHead=heading;%for scope graph
%IRHead=IRHeading;%for scope graph
yGain=bGain;% black
yIR=yReading;%blue
```

```
xOut=fix(xTest);% red
yOut=fix(yTest);%0;%fix(yTest);%orange
aGainOut=aGain;%track x over time
bGainOut=bGain;%track y over time
BotHead=heading;%for scope graph
IRHead=IRHeading;%for scope graph
IRSen01Dist=IR01;%For AHEAD_AVOID and/or FIND_SEEK; Used before ANNum calc of
2017-0207T
%2017-02-07T Used Global var. IRSen01Dist to calc ANNum later
%  in CHOICE BEHAVIOR modules EXPLORE, AVOID, SEEK
%
%Below IRSen01=IR01=IRSen01Dist passed to CONTEXT routine directly for tests.
IRSen01 = IR01;%set above in test loop
```

CONTEXT: Subsystem MATLAB function2

```
function [BEH,XPOS,YPOS]= Fitnessfcn(ctimein02,u)
%#codegen
%2017-0207T IR01=u(3) passsed from previous routine, and it
%  was used last fall 2016 for Dissertation ANSIM runs
%2015-12-05Sa Currently u values passed in are not used;
%only CueMoveStFBS is passed out (no scaling needed).
global HoldTimeAll dtimeall TotalTime;%keeps values between loops
global aGain bGain;%keeps values between loops
global tAvoidRun;%keeps values between loops
global tSeekWait;%keeps values between loops
global tSeekRun;%keeps value between loops
global AN01;%AN 0=off, 1=0n
global IRSen01Min IRSen01Max;%closest to object or wall

dtimeall=ctimein02-HoldTimeAll;%across all behaviors
TotalTime= TotalTime + dtimeall;
HoldTimeAll=ctimein02;
%Possibly: Reset this in behaviors, not in cue.%HoldTimeAll=
ctimein02;%across all behaviors

persistent ctimeprev;%keep value between loops
persistent CueMoveStFBSp tSpinWaitStart tForwStart;%keep value between loops
persistent tAvoidStart tAvoidRunFit;%
persistent tSpinDelay  tSpinRunStart;
persistent tSpinRun;
persistent IRSen01Test;%set test value with or w/o AN
persistent AheadFrac;%Reduce AvoidTime by some ratio

%coder.extrinsic('format','display');

if isempty(CueMoveStFBSp);
%if isempty(ctimeprev);
% ctimeprev=0.0;
 CueMoveStFBSp=0;
 tSpinWaitStart=ctimein02;%set below as: =ctimein02;%millis();%//times start
to wait before spin
 tSpinRunStart=ctimein02;%set below as: =ctimein02;%millis();
 tForwStart=ctimein02;
 tAvoidStart=ctimein02;
 tAvoidRunFit=tAvoidRun;%=0.125 %=0.2 %0.125;%0.125s
 tSpinDelay=tSeekWait;%=5.0s  %6.5;%6.5s
 tSpinRun=tSeekRun;%=0.5s %2.0;%2.0s
 AheadFrac=1;%total tAVOIDRunFit rotate time

 %test and set correct possible value ranges 2016-7-25; 4pm.
 if (IRSen01Min > 99);%should not occur
  IRSen01Min = 99;%keep in range
 elseif (IRSen01Max > 99);%should not occur
  IRSen01Max = 99;%keep in range
 elseif (IRSen01Min > IRSen01Max);
  IRSen01Max=  IRSen01Min;
 end;%if (IRSen01Min > 99);%should not occur

end;%if isempty(ctimeprev)
```

```
%moved next line here from below 2016-7-30A
IR01cm= u(3);%read from IR sensor as u(3); Added to Arduino Bot code


%%%%%%%%%%%%%%%%%%%  ANticipation additions %%%%%%%%%%%%%%%%%%%
 %set test value for distance with or w/o AN 2016-7-25; 4:30pm.
 if (AN01 > 0);%AN is on
  IRSen01Test= IRSen01Max;%wide range
 else;%AN01=0: AN off
  IRSen01Test= IRSen01Min;%narrow range
 end;%if (AN01 > 0);%AN is on

 %=============2nd set ANticipation additions ==============
 %From AVOID routine; shut off in AVOID: only here now
 %keep for block comment{
%Block comment line one above
%================================
%===AN should *not* be in AVOID; keep in Cue routine ML Function2
%===AVOID should just change heading; already cued to turn
%================================
%ANticipation: AN01:0=Off; 1=ON.
if (AN01 > 0);%ANticipation turned on.
 if (IR01cm== 100);%=  equal to max distance.
 %if (IR01cm~= 100);%~= not equal to max distance.
  AheadFrac=1;%minimum should never get here, but a protective test
 elseif (IR01cm<= IRSen01Min+1)
  AheadFrac=1;%minimum distance
 else
  %AheadFlag=1;
  AheadFrac= 1/(IR01cm-IRSen01Min);%Inverse relation, asymptote zero
  %AheadFrac= IRSen01Min/IR01cm;%40/IR01cm;%Base value= 40 set in Percepts
function
   %if (IR01cm < 40);%28);%//28cm=11in//25 cm=10in
     %2016-7-22F global variables handle in multiple routines.
     %Must also change in Percept function:  IRSen01fcn(u)
     %about line 31: IRDistMax=60;%28;%initial value %Setting as 28cm=11in.
closeness boundary to cue AVOID
     %Also change in AHEAD_AVOID function line 57 as base to divide into.
  AheadFrac=min(1,AheadFrac);% range <=1; should not get here, but protection
 end;%(AN01 > 0);%ANticipation turned on.
 %next 2 lines for places to stop in debug
 ARunTime2=ctimein02;%exit for loop
 Junk=1;
else
 AheadFrac=1;%Max AVOID turn time: tAvoidRun.
end;%(AN01 > 0);%ANticipation turned on.
%================================
%===AN should *not* be in AVOID; keep in Cue routine ML Function2
%===AVOID should just change heading; already cued to turn
%================================
%Block comment line2 follows
%keep for block comment}

 %=============END 2nd set ANticipation additions ==============

%%%%%%%%%%%%%%%  END ANticipation additions %%%%%%%%%%%%%%%%%%%%
```

```
%Next 2 lines not used in this function
%dtime= ctimein02 - ctimeprev;%time since last trip through loop (simulation,
not actual time)
%ctimeprev= ctimein02;%time now entered loop this time (simulation, not real
time)


%===code from Arduino:Nano20150608MvPWMSpin.t14ExtraForSLMLcode.txt
%===== void ChoiceBehavior()
%//*************************************************
%//*************************************************
%//Binary sketch size: 6,318 bytes (of a 30,720 byte maximum) 3:34 PM
9/1/2015

%%
%void ChoiceBehavior()
%...{
 %//Use to create function 6:10 PM 8/31/2015
 %//Cue Behavior: WAIT=0; EXPLORE=1; AVOID=2; SEEKCW=3; SEEKCCW=4; Back
only=5
 %keep these values for SL and Arduino Bot
 %//Cue OLD Move: Stop=0; Forward=1; Back=2; SpinCW=3; SpinCCW=4
 %//.............................
 if (ctimein02 < 1.0);%(millis() < 1000);%//1 sec
 %...{//WAIT before start
  CueMoveStFBSp= 0;%//Motion Forw, Back, Spin;
  %//Cue Behavior: WAIT=0; EXPLORE=1; AVOID=2; SEEKCW=3; SEEKCCW=4
  %keep these values for SL and Arduino Bot
  %//
  %//Test prints ++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  %Serial.print(CueMoveStFBSp);//Serial.print(" , ");
Serial.print(Sonar01cm);
  %Serial.println(" =CueMoveStFBSp; millis() < 1000");
  spCueMoveStFBSp= CueMoveStFBSp;   %print to ML Command Window
  spTotalTime= TotalTime; %print to ML Command Window
  %//
 %no end needed here because of else following. %...}//end first test if
(millis() < 1000)
 else
 %...{
  if (CueMoveStFBSp == 0)
  %...{
   tSpinWaitStart= ctimein02;%millis();%//times start to wait before spin
   tForwStart=ctimein02;%millis();
   CueMoveStFBSp= 1;
  end;%...}//EXPLORE start
  %...}//EXPLORE start
 %next line moved to end of SEEKfcn inclusive loop so not AVOID at start of
at time=0
 %end;%...}//end final if (millis() < 1000)
 %...}//end final if (millis() < 1000)
```

```
 %//=====================================================================
 %//AVOID only if AVOID started (CueMoveStFBSp= 3)
     % or EXPLORE current (CueMoveStFBSp= 1)
 %AVOID no longer interrupts SEEK
 if(CueMoveStFBSp== 1 || CueMoveStFBSp== 2);%(CueEXPLORE==1 || CueAVOID==2)




 %//.............................
 %moved next line above to use in ANticipation 2016-7-30A
 %IR01cm= u(3);%read from IR sensor as u(3); Added to Arduino Bot code
 if (IR01cm < IRSen01Test);%IRSen01Min);%40);%28);%//28cm=11in//25 cm=10in
  %above: AN01=0: IRSen01Min=IRSen01Test;  AN01=1: IRSen01Max=IRSen01Test
 %if (IR01cm < IRSen01Max);%IRSen01Min);%40);%28);%//28cm=11in//25 cm=10in
     %2016-7-22F global variables handle in multiple routines.
     %Must also change in Percept function:  IRSen01fcn(u)
     %about line 31: IRDistMax=60;%28;%initial value %Setting as 28cm=11in.
closeness boundary to cue AVOID
     %Also change in AHEAD_AVOID function line 57 as base to divide into.

 %...{//AVOID flag: Back & turn from wall

  if (CueMoveStFBSp ~= 2);%!= 2)//not equal to 2
  %...{
   tAvoidStart=ctimein02;%millis();
  end%...}end if (CueMoveStFBSp ~= 2)//AVOID start;

  CueMoveStFBSp= 2;%...//AVOID start;
  %//Cue Behavior: WAIT=0; EXPLORE=1; AVOID=2; SEEKCW=3; SEEKCCW=4

  %//......................
  if ( ((ctimein02-tAvoidStart) > tAvoidRunFit*AheadFrac) && (CueMoveStFBSp
== 2));%//2 sec; && is AND
  %if ( ((ctimein02-tAvoidStart) > tAvoidRunFit) && (CueMoveStFBSp ==
2));%//2 sec; && is AND
  %if ( ((millis()-tAvoidStart) > tAvoidRunFit) && (CueMoveStFBSp == 2));%//2
sec; && is AND
  %...{
   tSpinWaitStart= ctimein02;%millis();%//times start to wait before spin
   tForwStart=ctimein02;%millis();
   CueMoveStFBSp= 1;%//EXPLORE start;
   %//Cue Behavior: WAIT=0; EXPLORE=1; AVOID=2; SEEKCW=3; SEEKCCW=4
  end %...}//end if ( (ctimein02-tAvoidStart) > tAvoidRunFit)
 %...}//end if ( (millis()-tAvoidStart) > tAvoidRunFit)
 %//
  %//Test prints ++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  %Serial.print(CueMoveStFBSp);//Serial.print(" , ");
Serial.print(Sonar01cm);
  %Serial.println(" =CueMoveStFBSp; IR01cm < 25");
  spCueMoveStFBSp= CueMoveStFBSp;%print to ML Command Window
  %//
 %no end needed here since else follows. %...}//end first test if (IR01cm <
28 ) //25)
```

```
%.....................................................
 else %for this section, IR01cm > 28, so not near wall
     %else:  if (IR01cm <
IRSen01Test);%IRSen01Min);%40);%28);%//28cm=11in//25 cm=10in
 %...{
  if ( ((ctimein02-tAvoidStart) > tAvoidRunFit*AheadFrac) && (CueMoveStFBSp
== 2));% //2 sec; && is AND
  %if ( ((millis()-tAvoidStart) > tAvoidRunFit) && (CueMoveStFBSp == 2));%
//2 sec; && is AND
  %...{
   tSpinWaitStart= ctimein02;%millis();%//times start to wait before spin
   tForwStart=ctimein02;%millis();
   CueMoveStFBSp= 1;%//EXPLORE start; a
   %//Cue Behavior: WAIT=0; EXPLORE=1; AVOID=2; SEEKCW=3; SEEKCCW=4
  end %...}//end if ( (millis()-tAvoidStart) > tAvoidRunFit)
  %...}//end if ( (millis()-tAvoidStart) > tAvoidRunFit)

  %//
  %//Test prints ++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  %Serial.print(CueMoveStFBSp);//Serial.print(" , ");
Serial.print(Sonar01cm);
  %Serial.println(" =CueMoveStFBSp; IR01cm > 25");
  spCueMoveStFBSp= CueMoveStFBSp;%print to ML Command Window
  %//
  %//.............................
 end;%if (IR01cm < IRSen01Test);%IRSen01Min);%40);%28);%//28cm=11in//25
cm=10in
 %.....................................................
```

```
 %changes 2016-7-25M;4:30pm
 %end %...}//end final if (IR01cm < 28);//25)
 %...}//end final if (IR01cm < 28);//25)
 %checked to here 2015-11-16M, 1:15pm

 %//.............................
 Sonar01cm=u(2);
 if (Sonar01cm < 0.1);%0.1 as small value out of bounds %20)//20cm
 %...{//ANticipate: AVOID flag: Back & turn from wall
  %//need rule here
 end %...}//end if (Sonar01cm < 28)
 %...}//end if (Sonar01cm < 28)

end;%if (CueMoveStFBSp== 1 | CueMoveStFBSp== 2);  %(Cue EXPLORE || Cue  AVOID
)
%//=====================================================================


 %//.........................
```

```
 if (CueMoveStFBSp== 1);%//SEEK Spin start and test to continue/end
 %...{//EXPLORING timed for Spin of SEEK
 %//
  %//Test prints
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  %Serial.print(CueMoveStFBSp);Serial.print(" , "); Serial.print(millis()-
tSpinWaitStart);
  %Serial.println(" =CueMoveStFBSp; millis()-tSpinWaitStart; CueMoveStFBSp==
1");
  spCueMoveStFBSp=CueMoveStFBSp;%print to ML Command Window
  %//
   if ( (ctimein02-tSpinWaitStart) > tSpinDelay);%//1 sec
   %if ( (millis()-tSpinWaitStart) > tSpinDelay);%//1 sec
   %...{
   tSpinWaitStart=ctimein02;%millis();
   tSpinRunStart=ctimein02;%millis();
   CueMoveStFBSp= 3;% 3 for SLML, but 4 for Arduino Bot;%4;%//Motion Spin;
   %//Cue Behavior: WAIT=0; EXPLORE=1; AVOID=2; SEEKCW=3; SEEKCCW=4
   %no end needed here since else follows. %...}//end first if ( (millis()-
tSpinWaitStart> tSpinDelay)

  %//.............................
  else %//no Spin not EXPLOREing
  %...{
   CueMoveStFBSp= CueMoveStFBSp;%//no change to CueMoveStFBSp
   %//junk//let time pass by no update of start time: tSpinRunStart
   %//junk//tSpinWaitStart=millis();//Advance of start time to wait before
next SEEK Spin
  end%...}//end final if ( (ctimein02-tSpinWaitStart) > tSpinDelay)
  %...}//end final if ( (millis()-tSpinWaitStart) > tSpinDelay)
  %//**************************************



  %checked to here 2015-11-16M, 1:30pm
  %rechked to here 5:35pm, 2015-11-16M

 %no 'end' needed here since elseif follows. %...}//end first test if
(CueMoveStFBSp== 1)
 %//.........................
 %//....SEEK RUNS HERE..........SEEKRUNSHERE.............elseif,else,end
 %//.........................
 elseif (CueMoveStFBSp== 3);%3 for SLML, but 4 for Arduino Bot;%4);%//SEEK
Spin in Progress
 %...{
  if ( (ctimein02-tSpinRunStart) > tSpinRun);%//2.0 sec
  %if ( (millis()-tSpinRunStart) > tSpinRun);%//2.0 sec
  %...{

  %only if spin time has reached max as tSpinRun
   tSpinWaitStart=ctimein02;%millis();%//times start to wait before spin
   tSpinRunStart=ctimein02;%millis();
   tForwStart=ctimein02;%millis();
   CueMoveStFBSp= 1;%//EXPLORE again
   %//Cue Behavior: WAIT=0; EXPLORE=1; AVOID=2; SEEKCW=3; SEEKCCW=4
  else;%if ( (ctimein02-tSpinRunStart) > tSpinRun);: SEEK strill spinning
```

```
   %-------Check if AN01=1 is on ------------
   if (AN01 > 0);%AN is on
    %set above: IRSen01Test= IRSen01Max;%wide range
    if (IR01cm < IRSen01Max);%object in range of IRSen01Max
     %only if object in range of IRSen01Max
      tSpinWaitStart=ctimein02;%millis();%//times start to wait before spin
      tSpinRunStart=ctimein02;%millis();
      tForwStart=ctimein02;%millis();
      CueMoveStFBSp= 1;%//EXPLORE again
      %//Cue Behavior: WAIT=0; EXPLORE=1; AVOID=2; SEEKCW=3; SEEKCCW=4

    else;%if (IR01cm < IRSen01Max);%object in range of IRSen01Max
    %no change if no object in range of IRSen01Max.
    end;%if (IR01cm < IRSen01Max);%object in range of IRSen01Max

   else;%AN01=0: AN off
    %no change if AN01 not on.
   end;%if (AN01 > 0);%AN is on
   %-------end Check AN01=1 is on ------------

  end%...}//end if (ctimein02-tSpinRunStart) > tSpinRun)
  %...}//end if (millis()-tSpinRunStart) > tSpinRun)

 %no 'end' needed here since else follows. %...}//end second elseif
(CueMoveStFBSp== 3);%3 for SLML, but 4 for Arduino Bot;%4)
 %//.............................
 else
 %...{
  CueMoveStFBSp= CueMoveStFBSp;%//no change to CueMoveStFBSp
 end;%...}//end final if (CueMoveStFBSp== 1)
 %...}//end final if (CueMoveStFBSp== 1)
 %checked to here 1:45pm, 2015-11-16M
 %rechked to here 5:40pm, 2015-1-16M

 %%//
 %if ( (millis()-tPrint01Start) > tPrint01)
 %%...{
 % tTimeTrak01= millis()-tPrint01Start;%//time gap
 % tPrint01Start=millis();//reset time for printing
 % Serial.print(CueMoveStFBSp);Serial.print(" , ");
Serial.print(tTimeTrak01);
 % Serial.println(" =CueMoveStFBSp, tTimeTrak01, time=tPrint01");
 %end%...}//end first if ( (millis()-tPrint01Start) > tPrint01)
 %%...}//end first if ( (millis()-tPrint01Start) > tPrint01)
 %%//

%no 'end' needed here since ML entire function is in this call.%...}//end:
ChoiceBehavior()

%next line moved here from above WAIT set, so no start with AVOID at time=0
end;%...}//end final if (millis() < 1000)


%=====End Arduino code
```

```
%====Time flag examine code=========
if (ctimein02> 1.1 && ctimein02< 1.2) ;
 ctimein02;
end;%if (ctimein02> 17.4 && ctimein02< 18.1)
%====Time flag examine code=========

XPOS= aGain;%show on Scope13
YPOS= bGain;%show on Scope13
utotal= u(1)+u(2)+u(3);%weighted sum might not be used
%y = CueMoveStFBSp;%utotal;
if (ctimein02 == 20);% test of values at specific time
  CueMoveStFBSp;
  aGain;
  bGain;
end;%if (ctimein02 == 19);

BEH = CueMoveStFBSp;%utotal;
```

THRESHOLD COUPLING: Four potential choices

WAIT Behavior (Default): Pass through for Case =0
EXPLORE Behavior: Pass through for Case =1
AVOID Behavior: Pass through for Case =3
SEEK Behavior: Pass through for Case =4
Note for AHEAD and FIND Behaviors:
AHEAD: routine is part of the earlier code in CONTEXT function.
FIND routine is a specific situation of the SEEK Behavior.

CHOICE BEHAVIOR: only one of four possibilities is manifest or activated.
WAIT Behavior (Default): Case=0; Enabled Subsystem1/ML WAIT C0

```
function [da, db]= Waitfcn(ctimein4,u)%Code for WAIT
%#codegen
global aGain;
global bGain;%keeps values between loops
global daGain;
global dbGain;%keeps values between loops
global heading;
global speed;%keeps values between loops
%global variables initialized in Data Store Memory blocks in main SL
%   window.

global HoldTimeAll dtimeall TotalTime;%keeps values between loops
%see Fitnessfcn in Context section for dtimeall
%dtimeall=ctimein4-HoldTimeAll;%across all behaviors
%HoldTimeAll= ctimein4;%across all behaviors

%persistent dxy;%keeps values between loops
%persistent countpos;%keeps value between loops
daGain= dtimeall * 0;%dtimeall * speed * cos (heading);
dbGain= dtimeall * 0;%dtimeall * speed * sin (heading);
aGain= aGain +daGain;%Accumulate biomass/distance in 'a' direction.
bGain= bGain +dbGain;%Accumulate biomass/distance in 'b' direction.
da= daGain;
db= dbGain;
```

EXPLORE Behavior: Case=1; Enabled Subsystem2/ML EXPLORE C1

```
function [da, db]= EXPLOREfcn(ctimein1,u)
%EXPLORE: fitness cues choice to EXPLORE
%Sensor readings already taken & used to decide fitness.
global PrevBehavior;%know previous behavior for changes
global aGain;
global bGain;%keeps values between loops
global daGain;
global dbGain;%keeps values between loops
global heading;
global speed;%keeps values between loops
global AN01;%ANticipation: 0=OFF; 1=ON
global IRSen01Dist;%IR sensor 1 distance sensed
```

```
global IRSen01Min IRSen01Max;%closest to object or wall
global dANNum ANNumTot ANNum;%calc ANNum value; no commas between variables

global HoldTimeAll dtimeall TotalTime;%keeps values between loops
%see Fitnessfcn in Context section for dtimeall
%global variables initialized in Data Store Memory blocks in main SL
%   window.

coder.extrinsic('format','display', 'sprintf', 'strcat');

%persistent dtime1 HoldTime1;
%persistent dxy;%keeps values between loops
%persistent countpos;%keeps value between loops

%if isempty(dtime1);
% dtime1=0;
% HoldTime1=0;
%end

%dtime1= ctimein1-HoldTime1;%remove if not needed
%HoldTime1= ctimein1;%remove if not needed

%dtimeall=ctimein1-HoldTimeAll;%across all behaviors
%HoldTimeAll= ctimein1;%across all behaviors

daGain= dtimeall * speed * cos (heading);
dbGain= dtimeall * speed * sin (heading);
aGain= aGain +daGain;%Accumulate biomass/distance in 'a' direction.
bGain= bGain +dbGain;%Accumulate biomass/distance in 'b' direction.

PrevBehavior= 1;%EXPLORE now as PrevBehavior=1

   NH3= strcat (sprintf('in EXPLORE at %0.10f',ctimein1 ) );%concatenate only
2 strings at a time
   %NH3= strcat (ID2, sprintf(' =tNow01, at heading= %0.0f', spNR )
);%concatenate only 2 strings at a time
   disp(NH3) %displays without variable name

if (ctimein1 == 21);%check at specific time
    ctimein1;
end;% if (ctimein1 == 21);

%=========START ANNum calc add 2017-02-07T==============================

%++++++Set IRFactor distance to multiply by++++++
IRFactor=0;%init before IR value set for calc
if (AN01 > 0);%AN ON when AN01=1
 if (IRSen01Dist < IRSen01Max);
  IRFactor= IRSen01Dist;%<50cm
 else;
  IRFactor= IRSen01Max;%=50cm
 end;%if (IRSen01Dist < IRSen01Max);
else;%NO AN when AN01=0
 if (IRSen01Dist < IRSen01Min);
```

```
  IRFactor= IRSen01Dist;%<28cm
 else;
  IRFactor= IRSen01Min;%=28cm
 end;%if (IRSen01Dist < IRSen01Min);
end;%if (AN01 > 0);%AN ON when AN01=1
%++++++END Set IRFactor distance to multiply by++++++

% For EXPLORE:
% Area covered by IR beam is parallelogram: Ap= sin(IROffset) * b * h
% sin(IROffset)= sin(30 deg as default)= 0.5 for 30 deg default
 %for IROffset calc in PERCEPT as 30 default; need reset for any other angle
 % actually in ML IRSen01 value*
% b= IRFactor [was IRSen01Dist (global var.)];
% h= DistanceTraveled= sqrt (daGain*daGain + dbGain*dbGain);
%Thus the following equation:
dANNum= 0.5 * IRFactor * ( sqrt (daGain*daGain + dbGain*dbGain) );
%dANNum= 0.5 * IRSen01Dist * ( sqrt (daGain*daGain + dbGain*dbGain) );
ANNumTot= ANNumTot + dANNum;

if (ctimein1 > 0);
 ANNum= ANNumTot / ctimein1;
end;%if (ctimein1 > 0);

% Need ANNum output in OBSERVED RESPONSE section of Architecture
% actually in MATLAB Function1; when run is over at end time
%=========END ANNum calc add 2017-02-07T==================================

da= daGain;
db= dbGain;
```

AVOID Behavior: Case=2; Enabled Subsystem3/ML AVOID C2

```
function [daAvoid, dbAvoid, dheadAvoid,AheadFlag]= AVOIDfcn(ctimein2, cue4)
%SEEK: fitness cues choice to SEEK
%Sensor readings already taken & used to decide fitness.
global PrevBehavior;%know previous behavior for changes
global aGain;
global bGain;%keeps values between loops
global daGain;
global dbGain;%keeps values between loops
global heading;
%global speed;%keeps values between loops; Removed from GV list in this
module
%global tSeekWait;%keeps values between loops
global tAvoidStart;%keeps values between loops
global tAvoidRun;%keeps value between loops
global AVOIDAngle;%keeps value between loops
%global variables init in Data Store Memory blocks in main SL window.
global AN01;%ANticipation: 0=OFF; 1=ON
global IRSen01Dist;%IR sensor 1 distance sensed
global IRSen01Min IRSen01Max;%closest to object or wall
global dANNum ANNumTot ANNum;%calc ANNum value; no commas between variables
```

```
global HoldTimeAll dtimeall TotalTime;%keeps values between loops
%see Fitnessfcn in Context section for dtimeall
%dtimeall=ctimein3-HoldTimeAll;%across all behaviors
%HoldTimeAll= ctimein3;%across all behaviors

persistent HeadingStart AVOIDRateSpin;%keeps value between loops
%persistent AheadOn AheadFrac;%Flag & Reduce AvoidTime by some ratio
%persistent SEEKAngle;%keep values between loops

coder.extrinsic('format','display', 'sprintf', 'strcat');

AheadFlag=0;%Not used as of 2016-7-30A; keeps as SL in function output.

if isempty(HeadingStart);%initialize holdtime only
   HeadingStart=0.0;
   AVOIDRateSpin=0.0;
   daAvoid=0.0;
   dbAvoid=0.0;
   dheadAvoid=0.0;
   %AheadOn=0.0;
   %AheadFlag=0.0;
   %AheadFrac=1;
   %AVOIDAngle= (1/12)* 2 * pi()
   %2015-11-21Sat AVOIDAngle as global to include in Figure plot in EXPLORE
end

tNow01= ctimein2;%time of simulation
AVOIDRateSpin= AVOIDAngle/tAvoidRun;%rate of spin in radians

%next line for debug to stop at a specific time range.
if (tNow01 > 2.9 && tNow01 < 3.4);
%if (HoldTimeAll > 3 && HoldTimeAll < 4);
   ARunTime2=tNow01;%exit for loop
   Junk=1;
end;%(HoldTimeAll > 3 && HoldTimeAll < 4);

%{
%Block comment line one above
%================================
%===AN should *not* be in AVOID; keep in Cue routine ML Function2
%===AVOID should just change heading; already cued to turn
%================================
%ANticipation: AN01:0=Off; 1=ON.
if (AN01 > 0);%ANticipation turned on.
 if (IRSen01Dist== 100);%=  equal to max distance.
 %if (IRSen01Dist~= 100);%~= not equal to max distance.
  AheadFrac=1;%minimum should never get here, but a protective test
 elseif (IRSen01Dist<= IRSen01Min+1)
  AheadFrac=1;%minimum distance
 else
  AheadFlag=1;
  AheadFrac= 1/(IRSen01Dist-IRSen01Min);%Inverse relation, asymptote zero
  %AheadFrac= IRSen01Min/IRSen01Dist;%40/IRSen01Dist;%Base value= 40 set in
Percepts function
   %if (IR01cm < 40);%28);%//28cm=11in//25 cm=10in
```

```
     %2016-7-22F global variables handle in multiple routines.
     %Must also change in Percept function:  IRSen01fcn(u)
     %about line 31: IRDistMax=60;%28;%initial value %Setting as 28cm=11in.
closeness boundary to cue AVOID
     %Also change in AHEAD_AVOID function line 57 as base to divide into.
  AheadFrac=min(1,AheadFrac);% range <=1; should not get here, but protection
 end;%(AN01 > 0);%ANticipation turned on.
 %next 2 lines for places to stop in debug
 ARunTime2=tNow01;%exit for loop
 Junk=1;
else
 AheadFrac=1;%Max AVOID turn time: tAvoidRun.
end;%(AN01 > 0);%ANticipation turned on.
%================================
%===AN should *not* be in AVOID; keep in Cue routine ML Function2
%===AVOID should just change heading; already cued to turn
%================================
%Block comment line2 follows
%}

%if (tAvoidStart == 0  );%AVOID not started yet, start this time.
%if (PrevBehavior ~= 2  );%~= means not equal (see ML relational operators
if (PrevBehavior ~= 2 || (PrevBehavior == 2 && tAvoidStart==0) );%~= means
not equal (see ML relational operators
  %first time to run SEEK as Behavior 3; restart timer
  PrevBehavior=2;%reset PrevBehavior
  tAvoidStart = tNow01;%set tSeekStart time
  HeadingStart = heading;
else%continue
end;%if (PrevBehavior ~= 2);

%replaced by above test: if (PrevBehavior ~= 3)
%if (tSeekStart == 0.0);%first time to run SEEK
%  tSeekStart = tNow01;%set tSeekStart time
%  HeadingStart = heading;
%else%continue
%end;%if (tSeekStart == 0.0)

%=====AVOID Spin to (1/12)* 2 * pi()= 15 deg =================
%Test on next line not needed in AVOID
%if ( (tNow01 - tAvoidStart) < tAvoidRun * AheadFrac )%SEEK running
 heading= heading + dtimeall * AVOIDRateSpin  ;
 difHead= dtimeall * AVOIDRateSpin;

   spNR= heading;
   ID= sprintf('%0.3f', tNow01);
   %ID= sprintf('%03d', tNow01);
   ID2= strcat (ID, char(58) );%add a colon: as char(58)
   %http://www.mathworks.com/help/matlab/ref/char.html?searchHighlight=char
   %NH= sprintf('%0.0f', xlayout, spNR )
   %NH2= strcat (ID, sprintf('%0.0f', spNR ) )
   %formatSpec = 'The array is %dx%d.';
   %A1 = 2;
   %A2 = 3;
   %str = sprintf(formatSpec,A1,A2)
```

```
   NH3= strcat (ID2, sprintf(' =tNow01, AVOID at heading= %0.3f', spNR )
);%concatenate only 2 strings at a time
   %NH3= strcat (ID2, sprintf(' =tNow01, at heading= %0.0f', spNR )
);%concatenate only 2 strings at a time
   disp(NH3) %displays without variable name

   PrevBehavior=2;%reset PrevBehavior

%else
 %tAvoidStart = 0.0;%reset to wait for next SEEK start
 %difHead= 0.0;%no change in heading
 %PrevBehavior=2;%reset PrevBehavior

%Test end on next line not needed in AVOID
%end;%if ( (tNow01 - tAvoidStart) < tAvoidRun )

%=========Adjust -2*pi()<heading< 2*pi()===================
if (heading < -2*pi() );%low range of heading
  heading= heading + 2*pi();%
elseif (heading > 2*pi() );%high range of heading
  heading= heading - 2*pi();%
else%continue: in range
end;%if (heading < -2*pi() );

%========no move Forward/Back ============================
daGain= dtimeall * 0;%no change; %was: dtimeall * speed * cos (heading);
dbGain= dtimeall * 0;%no change; %was: dtimeall * speed * sin (heading);
aGain= aGain +daGain;%Accumulate biomass/distance in 'a' direction.
bGain= bGain +dbGain;%Accumulate biomass/distance in 'b' direction.

PrevBehavior= 2;%AVOID now as PrevBehavior=2

%=========START ANNum calc add 2017-02-07T==============================

%++++++Set IRFactor distance to multiply by++++++
IRFactor=0;%init before IR value set for calc
if (AN01 > 0);%AN ON when AN01=1
 if (IRSen01Dist < IRSen01Max);
  IRFactor= IRSen01Dist;%<50cm
 else;
  IRFactor= IRSen01Max;%=50cm
 end;%if (IRSen01Dist < IRSen01Max);
else;%NO AN when AN01=0
 if (IRSen01Dist < IRSen01Min);
  IRFactor= IRSen01Dist;%<28cm
 else;
  IRFactor= IRSen01Min;%=28cm
 end;%if (IRSen01Dist < IRSen01Min);
end;%if (AN01 > 0);%AN ON when AN01=1
%++++++END Set IRFactor distance to multiply by++++++

% For both AHEAD_AVOID and SEEK when AN01=1:
% Area covered by entire IR beam is triangle: At= 0.5 * b * h
% b= IRFactor [was IRSen01Dist (global var.)];
% h= sin(difHead) * IRSen01Dist; should work for small angles
```

```
%Thus the following equation:
dANNum= 0.5 * IRFactor * ( sin(difHead) * IRFactor);
%dANNum= 0.5 * IRSen01Dist * ( sin(difHead) * IRSen01Dist);
ANNumTot= ANNumTot + dANNum;

if (ctimein2 > 0);
 ANNum= ANNumTot / ctimein2;
end;%if (ctimein2 > 0);

% Need ANNum output in OBSERVED RESPONSE section of Architecture
% actually in MATLAB Function1; when run is over at end time
%=========END ANNum calc add 2017-02-07T================================

spctimein2= ctimein2;
daAvoid= daGain;
dbAvoid= dbGain;
dheadAvoid= difHead;
%AheadFlag=AheadOn;
spdheadAvoid= dheadAvoid;
```

SEEK Behavior: Case=3; Enabled Subsystem4/ML SEEK C3

```
function [da, db, dhead]= SEEKfcn(ctimein3, u)
%SEEK: fitness cues choice to SEEK
%Sensor readings already taken & used to decide fitness.
global PrevBehavior;%know previous behavior for changes
global aGain;
global bGain;%keeps values between loops
global daGain;
global dbGain;%keeps values between loops
global heading;
global speed;%keeps values between loops
global tSeekWait;%keeps values between loops
global tSeekStart;%keeps values between loops
global tSeekRun;%keeps value between loops
global SEEKAngle;%keeps value between loops
global AN01;%ANticipation: 0=OFF; 1=ON
global IRSen01Dist;%IR sensor 1 distance sensed
global IRSen01Min IRSen01Max;%closest to object or wall
global dANNum ANNumTot ANNum;%calc ANNum value; no commas between variables
%global variables init in Data Store Memory blocks in main SL window.

global HoldTimeAll dtimeall TotalTime;%keeps values between loops
%see Fitnessfcn in Context section for dtimeall
%dtimeall=ctimein3-HoldTimeAll;%across all behaviors
%HoldTimeAll= ctimein3;%across all behaviors

coder.extrinsic('format','display', 'sprintf', 'strcat');

persistent HeadingStart RateSpin;%keeps value between loops
%persistent SEEKAngle;%keep values between loops
```

```
if isempty(HeadingStart);%initialize holdtime only
   HeadingStart=0;
   RateSpin=0.0;
   %SEEKAngle= (3/4)* 2 * pi();%(3/4)*pi();%SEEK total spin angle
   %2015-11-5R SEEKAngle as global to include in Figure plot in EXPLORE
end

tNow01= ctimein3;%time of simulation
RateSpin= SEEKAngle/tSeekRun;%rate of spin in radians

%coder.extrinsic('get_param');%function defined in ML (external to program).
%coder.extrinsic('get_param','getSimulinkBlockHandle');

%tNow= get_param('YourModel','SimulationTime');
%double tNow01= get_param('ANSimDelay20151102M','SimulationTime');%name of SL
file
%BlockHandle = getSimulinkBlockHandle('vdp/Fcn',true);
%tNow01= get_param(BlockHandle,'SimulationTime');

if (PrevBehavior ~= 3);%~= means not equal (see ML relational operators
  %first time to run SEEK as Behavior 3; restart timer
  PrevBehavior=3;%reset PrevBehavior
  tSeekStart = tNow01;%set tSeekStart time
  HeadingStart = heading;
else%continue
end;%if (PrevBehavior ~= 3);

%replaced by above test: if (PrevBehavior ~= 3)
%if (tSeekStart == 0.0);%first time to run SEEK
%  tSeekStart = tNow01;%set tSeekStart time
%  HeadingStart = heading;
%else%continue
%end;%if (tSeekStart == 0.0)

%=====Spin to (3/4)* 2 * pi()=================
if ( (tNow01 - tSeekStart) < tSeekRun )%SEEK running
 heading= heading + dtimeall * RateSpin  ;
 difHead= dtimeall * RateSpin;
 %---------old version without dtimeall
 %{
 prevHeading= heading;
 deltaHead= SEEKAngle * ((tNow01-tSeekStart)/tSeekRun);
 %heading= HeadingStart +  ( (3.0*pi()/4) * ((tNow-tSeekStart)/tSeekRun) );
 heading= HeadingStart +  deltaHead;%may allow for ML variable time step
 difHead= heading- prevHeading;
 %}
 %--------end old version without dtimeall

else
 tSeekStart = 0.0;%reset to wait for next SEEK start
 difHead= 0.0;%no change in heading
end;%if ( (tNow - tSeekStart) < tSeekRun )

%=========Adjust -2*pi()<heading< 2*pi()==================
if (heading < -2*pi() );%low range of heading
```

```
  heading= heading + 2*pi();%
elseif (heading > 2*pi() );%high range of heading
  heading= heading - 2*pi();%
else%continue: in range
end;%if (heading < -2*pi() );

%========no move Forward/Back ==========================
daGain= dtimeall * 0;%no change; %was: dtimeall * speed * cos (heading);
dbGain= dtimeall * 0;%no change; %was: dtimeall * speed * sin (heading);
aGain= aGain +daGain;%Accumulate biomass/distance in 'a' direction.
bGain= bGain +dbGain;%Accumulate biomass/distance in 'b' direction.

PrevBehavior= 3;%SEEK now as PrevBehavior=3

spctimein3= ctimein3;
   NH3= strcat (sprintf('in SEEK at %0.10f',ctimein3 ) );%concatenate only 2
strings at a time
   %NH3= strcat (ID2, sprintf(' =tNow01, at heading= %0.0f', spNR )
);%concatenate only 2 strings at a time
   disp(NH3) %displays without variable name

%nmot work: disp ('in SEEK at %0.10f',ctimein3);%shows ctimein3 without
variable name, only value show.
%disp (ctimein3);%shows ctimein3 without variable name, only value show.

%=========START ANNum calc add 2017-02-07T==============================

%++++++Set IRFactor distance to multiply by++++++
IRFactor=0;%init before IR value set for calc
if (AN01 > 0);%AN ON when AN01=1
 if (IRSen01Dist < IRSen01Max);
  IRFactor= IRSen01Dist;%<50cm
 else;
  IRFactor= IRSen01Max;%=50cm
 end;%if (IRSen01Dist < IRSen01Max);
else;%NO AN when AN01=0
 if (IRSen01Dist < IRSen01Min);
  IRFactor= IRSen01Dist;%<28cm
 else;
  IRFactor= IRSen01Min;%=28cm
 end;%if (IRSen01Dist < IRSen01Min);
end;%if (AN01 > 0);%AN ON when AN01=1
%++++++END Set IRFactor distance to multiply by++++++

% SEEK IR beam does not respond during spin for NO AN
%  thus, only count coverage for AN01==1
% For both AHEAD_AVOID and SEEK when AN01==1:
if (AN01==1);%only cover area when AN is ON, for AN01==1.
% Area covered by entire IR beam is triangle: At= 0.5 * b * h
% b= IRFactor [was IRSen01Dist (global var.)];
% h= sin(difHead) * IRSen01Dist; should work for small angles
%Thus the following equation:
dANNum= 0.5 * IRFactor * ( sin(difHead) * IRFactor);
%dANNum= 0.5 * IRSen01Dist * ( sin(difHead) * IRSen01Dist);
ANNumTot= ANNumTot + dANNum;
```

```
if (ctimein3 > 0);
  ANNum= ANNumTot / ctimein3;
 end;%if (ctimein3 > 0);
end;%if (AN01==1);%only cover area when AN is ON, for AN01==1.

% Need ANNum output in OBSERVED RESPONSE section of Architecture
% actually in MATLAB Function1; when run is over at end time
%=========END ANNum calc add 2017-02-07T================================

da= daGain;
db= dbGain;
dhead= difHead;
```

OBSERVED RESPONSE: Subsystem MATLB Function1

```
function [y, theading, tANNum, EXPLORENumOut, AVOIDNumOut, SEEKNumOut]=
Behsfcn(Beh02)
%#codegen
%2016-5-9M 6:35pm
%size of NicheLayout:  zeros(52,52,3)

global heading;
global StopAtTime aGain bGain;%keeps value between loops
global HoldTimeAll dtimeall TotalTime;%keeps values between loops
global NicheLayout;%from NICHE section ML function
global xReadingFixPlot yReadingFixPlot;%Plot IR distance boundary
global xReadingMinPlot yReadingMinPlot;%Plot Min IR distance boundary
global AN01;%AN 0=off, 1=0n
global IRSen01Dist;%IR sensor 1 distance sensed
global dANNum ANNumTot ANNum;%calc ANNum value; no commas between variables

%see Fitnessfcn in Context section for dtimeall
persistent BehM6xPts;%keeps value between loops
persistent Beh5xPts;%keeps value between loops: final print to ML  Command
Window .
persistent Pos3xPts Pos2xPts Pos2xPts10th;%position values retained b/t loops
persistent IREnd01 IREndMin;%Max & Min Ends of IR reading beam retained
between loops.
persistent count01;%keeps value between loops
persistent HoldBeh;%keeps value between loops
persistent IRPathBotMinDist IRPathBotMaxDist;
persistent EXPLORENum AVOIDNum SEEKNum uNum;

coder.extrinsic('format','display','annotation', 'sprintf', 'strcat');
% coder.extrinsic('format','display', 'sprintf', 'strcat');

nPtsMax=160;
n10PtsMax=1+nPtsMax/10;

if isempty(BehM6xPts);
 BehM6xPts=zeros(8,nPtsMax);%2016-7-21R;size increased to 8 vs. 6
 Beh5xPts=zeros(7,nPtsMax);%2016-7-21R;size increased to 7 vs. 5
 Pos3xPts=zeros(3,nPtsMax);
 Pos2xPts=zeros(2,nPtsMax);
 IREnd01 =zeros(2,nPtsMax);%2016-09-23F Plot end of IR sense reading
 IREndMin =zeros(2,nPtsMax);%2016-09-26M Plot Min end of IR sense reading
 Pos2xPts10th=zeros(2,n10PtsMax);
 IRPathBotMinDist=zeros(2,3*nPtsMax);%2016-09-26M Plot IRPath bot to min dist
 IRPathBotMaxDist=zeros(2,3*nPtsMax);%2016-09-23F Plot IRPath bot to max dist
  count01=0;
 HoldBeh=-1;
 EXPLORENum=0;
 AVOIDNum=0;
 SEEKNum=0;
 uNum= [0 0 0];
 uNumOut= [0 0 0];%not used as of 2017-2-9R
 EXPLORENumOut=0;
 AVOIDNumOut=0;
 SEEKNumOut=0;
```

```
end;%if isempty(BehM6xPts);

%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+

%======ANNum for each Behavior: EXPLORENum, AVOIDNum, SEEKNum
%==passed in: Beh02(1to6)= [clocktime clocktime  CueEXP CueAVOID CueSEEK
CueWAIT CueAHEAD CueAHEAD]
if (Beh02(3)==1);%Beh02(3) is CueEXPLORE:0=off; 1=ON
 EXPLORENum= EXPLORENum + dANNum;
elseif (Beh02(4)==2);%Beh02(4) is CueAVOID:0=off; 2=ON
 AVOIDNum= AVOIDNum + dANNum;
elseif (Beh02(5)==3);%Beh02(5) is CueSEEK:0=off; 3=ON
 if (AN01==1);%when AN is ON: AN01==1
  %only change in SEEKNum when AN is ON: AN01==1
  SEEKNum= SEEKNum + dANNum;
 else;%when NO AN: AN01==0
   %No change in SEEKNum
 end;%if (AN01==1);%when AN is ON: AN01==1
else
 %no cue of EXPLORE, AVOID, or SEEK. Possibly WAIT or AHEAD
 %WAIT: no accummulation of dANNum
 %AheadFlag not used in AHEAD_AVOID as of 2017-2-9R
end;%if (Beh02(3)==1);%Beh02(3) is CueEXPLORE:0=off; 1=ON
uNum=[ EXPLORENum AVOIDNum SEEKNum ] ;%pass out for plot

%======END ANNum for each Behavior: EXPLORENum, AVOIDNum, SEEKNum

%Setup XYGraph in EXPLORE subsystem
%http://www.mathworks.com/help/matlab/creating_plots/access-and-modify-
property-values.html
%already declared above: global StopAtTime;
 persistent holdtime2;
 coder.extrinsic('format','display')

 if isempty(holdtime2);%initialize holdtime only
   holdtime2=0;
 end
%set(0,'ShowHiddenHandles','on')
%set(gcf,'menubar','figure','toolbar','none');%add menubar to the figure
set(gcf,'menubar','none','toolbar','none')%no menubar or toolbar on figure
ctime2= Beh02(1);%u(1);%time from clock
%ctime= u(1);%time from clock

%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+

%next line for debug to stop at a specific time range.
if (HoldTimeAll > 2.9 && HoldTimeAll < 3.4);
%if (HoldTimeAll > 18.7 && HoldTimeAll < 20); 6*pi
%if (HoldTimeAll > 3 && HoldTimeAll < 4);
   ARunTime2=HoldTimeAll;%exit for loop
   StopAtTime;
   holdtime2;
   ctime2;
```

```
   Junk=1;
end;%(HoldTimeAll > 3 && HoldTimeAll < 4);

%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+

%================================
%NOT WORKING 2016-07-31U, 3:05pm
%WORKING: 2015-10-27T, 3:30pm
%NOT WORKING as of 2015-10-26M, 6:30PM

%choose only samples on each second
%ceil(x) == floor(x) from
%https://www.mathworks.com/matlabcentral/newsreader/view_thread/163080
%Persistent example
%http://stackoverflow.com/questions/16097708/storing-data-in-matrix-format-
in-simulink
%persistent mat
%if isempty(mat)
%  mat = zeros(10,2);
%  cnt = 1; % Counter to count number of times enabled
%end
%
%if cnt <= 10
%  mat(cnt,1) = A;
%  mat(cnt,2) = B;
%end
%cnt = cnt + 1;
BehSize01= size(Beh02);
Pos3xPtssize= size(Pos3xPts);

%for ncol1   = 1: 1: BehSize01(2);%check all cols
 %prtncol1= ncol1;
 %prtflrBeh6X190intrans=floor(Beh6X190intrans(1,ncol1))
  %floor rounds a number to the next smaller integer.
 if Beh02(1)==0;%1st row, 1st col tested
  count01=count01+1;

  %.........................
  %limit of matrices ranges
  if count01 > nPtsMax;
   count01= nPtsMax;%limit of matrix index
  end;%if count01> nPtsMax;%20;
  %.........................

  BehM6xPts(:,count01)= Beh02;%first col
  HoldBeh= Beh02(1);% keep value to test against next loop
  %Place Positions into matrix
  %from file: ANSimandML20150225W.m
  %A = [5 7 8;
  %     0 1 9;
  %     4 3 6];
  %HoldPos= [  Beh02(1) ; aGain ; bGain ]
  Pos3xPts(:,count01)= [  Beh02(1) ; aGain ; bGain ];
  IREnd01(:,count01)= [  xReadingFixPlot ; yReadingFixPlot ];
  IREndMin(:,count01)= [  xReadingMinPlot ; yReadingMinPlot ];
```

```
  %below works 2 lines
  %HoldPos= [  Beh02(1) ; aGain ; bGain ]
  %Pos3xPts(:,count01)= HoldPos;
 elseif (floor(Beh02(1)) == Beh02(1)) && (HoldBeh < Beh02(1));
  count01=count01+1;

  %.........................
  %limit of matrices ranges
  if count01 > nPtsMax;
   count01= nPtsMax;%limit of matrix index
  end;%if count01> nPtsMax;%20;
  %.........................

  BehM6xPts(:,count01)= Beh02;%Error Here; each col
  %floor rounds a number to the next smaller integer.
  HoldBeh= Beh02(1);% keep value to test against next loop
  %Place Positions into matrix
  Pos3xPts(:,count01)= [  Beh02(1) ; aGain ; bGain ];
  IREnd01(:,count01)= [  xReadingFixPlot ; yReadingFixPlot ];
  IREndMin(:,count01)= [  xReadingMinPlot ; yReadingMinPlot ];

 elseif (Beh02(1)> (StopAtTime-0.01) ) && (HoldBeh < StopAtTime);%max run
time=18.8496 = 6*pi()
 %elseif (Beh02(1)> (StopAtTime-0.0001) ) && (HoldBeh < StopAtTime);%max run
time=18.8496 = 6*pi()
 %elseif (Beh02(1)> 9.4247) && (HoldBeh < 9.4247);%max run time=18.8496 =
6*pi()
 %elseif (Beh02(1)> 18.81) && (HoldBeh < 18.81);%max run time=18.8496 =
6*pi()
  %count01=count01+1;

  %Use below 7 lines code if StopAtTime < 20; such as 6*pi
  %comment out section
  %{
  BehM6xPts(:,20)= Beh02;%each col
  HoldBeh= Beh02(1);% keep value to test against next loop
  BehM6xPts(1,20)= floor(BehM6xPts(1,20)+1);%each col
  %Place Positions into matrix
  Pos3xPts(:,20)= [  Beh02(1) ; aGain ; bGain ];
  %Pos3xPts(1,20)= 19;
  Pos3xPts(1,20)= floor(Pos3xPts(1,20)+1);
  %}
  %end of comment out section

  format('short');%see above: coder.extrinsic('format','display')
  Beh5xPts=BehM6xPts( [1 3 4 5 6 7 8], :)%output to ML Command Window
  format('bank');%round to 2 digits right of decimal
  Pos2xPts=Pos3xPts( [2 3], :)%output to ML Command Window
  format('bank');%round to 2 digits right of decimal
  B5X20P2X20=cat(1, Beh5xPts,  Pos2xPts ) %output to ML Command Window

  spTotalTime=TotalTime %should agree with StopAtTime (of prog) & StopTime
(of SL)

 else %no action
```

```
 end%if Beh6X190intrans(1)==0;%first col tested

%Beh5X19=Beh1scol( [1 3 4 5 6], :);
%Beh4X19=Beh1scol( [  3 4 5 6], :)
%Next line needed for reference; see aabove in elseif
%Beh5xPts=BehM6xPts( [1 3 4 5 6], :)%moved to elseif for single output to
Command Window

%=========Figure inserted here from EXPLORE============================
% Graph below=============================================
%http://www.mathworks.com/help/matlab/creating_plots/access-and-modify-
property-values.html
%if (ctime2 >= 18.8  );% global end is StopTime;pi()*6=9.4248

%Major length of simulation note:
%in Simulink diagram, upper left, set StopAtTime=n*pi to match value of n*pi
at top of simuation window.
%This will automatically output graph at end of simuation runtime:
StopAtTime

if (ctime2 > (StopAtTime-0.01)  )&& (holdtime2<ctime2) && (ctime2 <
StopAtTime+0.0001);% global end is StopTime;pi()*6=9.4248
%if (ctime2 > (StopAtTime-0.01)  )&& (holdtime2<ctime2) && (ctime2 <
StopAtTime+0.0001);% global end is StopTime;pi()*6=9.4248
%if (ctime2 > (StopAtTime-0.0001)  )&& (holdtime2<ctime2) && (ctime2 <
StopAtTime+0.0001);% global end is StopTime;pi()*6=9.4248
%if (ctime > 9.4247)&& (holdtime<ctime) && (ctime < 9.4249);% global end is
StopTime;pi()*6=9.4248
%if (ctime > 18.8495)&& (holdtime<ctime) && (ctime < 18.8497);% global end is
StopTime; pi()*6=18.8496

 for iFill = count01: 1 : nPtsMax;%Fill zeros of matrix with end point
reached
   %Pos2xPts=Pos3xPts( [2 3], :)%output to ML Command Window [from above]
   %Pos2xPts(:, iFill )= [ aGain, bGain];
   Pos2xPts(:, iFill )= [ Pos2xPts(1, count01 ), Pos2xPts(2, count01 )];%Fill
Matrix to end with last point
   IREnd01(:, iFill )= [ IREnd01(1, count01 ) , IREnd01(2, count01 )];%Fill
Matrix to end with last point
   IREndMin(:, iFill )= [ IREndMin(1, count01 ) , IREndMin(2, count01
)];%Fill Matrix to end with last point

 end;%for iFill = count01: 1 : nPtsMax;

 %-------------Find points to graph every 10th point-------------
  for i10Pts = 10: 10 : nPtsMax;%Fill zeros of matrix with end point reached
   %Pos2xPts=Pos3xPts( [2 3], :)%output to ML Command Window [from above]
   %Pos2xPts(:, iFill )= [ aGain, bGain];
   if (i10Pts > nPtsMax);
       break;
   end;%(i10Pts > n10PtsMax);
   %if(i10pts == 0);%start index at 1, not zero.
   % Pos2xPts10th(:, 1 )=[ Pos2xPts(1, 1 ), Pos2xPts(2, 1 )];
   %else
```

```
   % Pos2xPts10th(:, (i10Pts/10) )=[ Pos2xPts(1, i10Pts ), Pos2xPts(2, i10Pts
)];
   %end;%(i10pts == 0);
   Pos2xPts10th(:, (i10Pts/10) )=[ Pos2xPts(1, i10Pts ), Pos2xPts(2, i10Pts
)];
  end;%for i10Pts = 0: 10 : nPtsMax;
  Pos2xPts10th(:, n10PtsMax )=[ Pos2xPts(1, nPtsMax ), Pos2xPts(2, nPtsMax
)];
  Pos2xPts10th %print every 10th pt.
 %-------------end Find points to  graph every 10th point----------

 IREnd01 %print IR end points
 IREndMin %print IR Min beam end points


 %-------------Gen path bot to min distance-------------
  for iIRminPts = 0 : 3 : 3*nPtsMax;%Join all points together
   %Pos2xPts=Pos3xPts( [2 3], :)%output to ML Command Window [from above]
   %Pos2xPts(:, iFill )= [ aGain, bGain];
   if (iIRminPts > 3*nPtsMax);
       break;
   end;%(i10Pts > n10PtsMax);
   if (  ((iIRminPts/3)+1) > nPtsMax);
       break;
   end;%(i10Pts > n10PtsMax);
   %if(i10pts == 0);%start index at 1, not zero.
   % Pos2xPts10th(:, 1 )=[ Pos2xPts(1, 1 ), Pos2xPts(2, 1 )];
   %else
   % Pos2xPts10th(:, (i10Pts/10) )=[ Pos2xPts(1, i10Pts ), Pos2xPts(2, i10Pts
)];
   %end;%(i10pts == 0);
   %Pos2xPts10th(:, (i10Pts/10) )=[ Pos2xPts(1, i10Pts ), Pos2xPts(2, i10Pts
)];

   %IR Max dist
   IRPathBotMaxDist(:, iIRminPts+1 ) = Pos2xPts(:, (iIRminPts/3)+1  );
   IRPathBotMaxDist(:, iIRminPts+2 ) = IREnd01( :, (iIRminPts/3)+1  );
   IRPathBotMaxDist(:, iIRminPts+3 ) = Pos2xPts(:, (iIRminPts/3)+1  );
   %IR min distance
   IRPathBotMinDist(:, iIRminPts+1 ) = Pos2xPts(:, (iIRminPts/3)+1  );
   IRPathBotMinDist(:, iIRminPts+2 ) = IREndMin( :, (iIRminPts/3)+1  );
   IRPathBotMinDist(:, iIRminPts+3 ) = Pos2xPts(:, (iIRminPts/3)+1  );

  end;%for iIRminPts = 1: 1 : nPtsMax;%Join all points together
  %Pos2xPts10th(:, n10PtsMax )=[ Pos2xPts(1, nPtsMax ), Pos2xPts(2, nPtsMax
)];
  IRPathBotMinDist %print every Min pt.
  IRPathBotMaxDist %print every Max pt.
 %-------------end Gen path bot to min distance----------

 holdtime2= ctime2;
 showu=Beh02;

    FName01= strcat (sprintf('%0.0f =ANNum, INTEGRATED PATH',ANNum )
);%concatenate only 2 strings at a time
```

```
   %FName01= strcat (sprintf('INTEGRATED PATH, ANNUM= %0.0f',ANNum )
);%concatenate only 2 strings at a time
   %NH3= strcat (sprintf('in EXPLORE at %0.10f',ctimein1 ) );%concatenate
only 2 strings at a time
   %NH3= strcat (ID2, sprintf(' =tNow01, at heading= %0.0f', spNR )
);%concatenate only 2 strings at a time
   disp(FName01) %displays without variable name

 NameFig1= FName01;
 %NameFig1= 'INTEGRATED PATH, v=15cm/s';
 if count01 == nPtsMax;
    NameFig1= 'INTEGRATED PATH, Max # Points';
 end;%if count01 == nPtsMax;

 figure1 = figure('Tag','SIMULINK_XYGRAPH_FIGURE','NumberTitle','off',...
    'Name',NameFig1,...
    'IntegerHandle','off');
%    'Name','INTEGRATED PATH, v=2cm/s',...
%fig = gcf %returns the current figure handle. If a figure does not exist,
then gcf creates a figure and returns its handle.
% Create axes
%axes1 = axes(figure1,...  %not work
axes1 = axes('Parent',figure1,...
    'Position',[0.150 0.1500 0.81 0.7300]);%'Position',[0.130 0.2000 0.86
0.7900]);
%    'Position',[0.11437908496732 0.178082191780822 0.853223506825338
0.718144223313517]);
%Position – Location and size of figure's drawable area: [left bottom width
height]
%    'OuterPosition',[0.120 0.1900 0.96 0.8500],...
%OuterPosition – Location and size of figure's outer bounds: [left bottom
width height]

xPosMin= min(Pos2xPts(1,:)) - .1*abs(min(Pos2xPts(1,:))) - 1;
xPosMax= max(Pos2xPts(1,:)) + .1*abs(max(Pos2xPts(1,:))) + 1;
yPosMin= min(Pos2xPts(2,:)) - .1*abs(min(Pos2xPts(2,:))) - 1;
yPosMax= max(Pos2xPts(2,:)) + .1*abs(max(Pos2xPts(2,:))) + 1;

% Uncomment the following 2 lines for X-limits & Y-limits of the axes
 xlim(axes1,[xPosMin xPosMax]);%[-4 5]);
 ylim(axes1,[yPosMin yPosMax]);%[-4 5]);
% Uncomment next 2 lines to set specific axis range
 xlim(axes1,[-5 110]);
 ylim(axes1,[-5 110]);
 %xlim(axes1,[0 101]);
 %ylim(axes1,[0 101]);

 %xlim(axes1,[-150 350]);%[-4 5]);
 %ylim(axes1,[-0 320]);%[-4 5]);

%first plot line originally here
%line01 = line('XData',Pos2xPts(1,:),'YData',Pos2xPts(2,:),...
%       'MarkerSize', 8 , 'LineWidth', 2 ,...
%       'Marker','square','color','black');
```

```
%next line from EXPLORE
%line01 = line('XData',abGain3X20(2,:),'YData',abGain3X20(3,:),...
%        'Marker','square','color','black');

% Create xlabel & ylabel & title
xlabel('DISTANCE (x)');
ylabel('DISTANCE (y)');
%xlabel('BIOMASS/DIST (x)');
%ylabel('BIOMASS/DIST (y)');
title('LOCATION');


hold on;
%-----------IR Sense End Pts----------------
%line03 = line('XData',IREnd01(1,:),'YData',IREnd01(2,:),...
%              'LineStyle',':','MarkerFaceColor', 'red',...
%              'MarkerSize', 3 , 'LineWidth', 2 ,...
%              'Marker','o','color','red');%'LineStyle',':', makes dots
%
  %2016-9-23F Error Caused by:
  %While setting the 'Marker' property of 'Line': 'circle' is not a valid
value.
  %Use one of these values: '+' | 'o' | '*' | '.' | 'x' | 'square' |
'diamond' | 'v' | '^' | '>' | '<' | 'pentagram' | 'hexagram' | 'none'.
%-----------end IR Sense End Pts----------------

%%%%%%%%%%IRPathBotMaxDist;%%%%%%%%%%%%%%%%%%%%%%%%%%%%
hold on;
%-----------IRPathBotMaxDist----------------
line04 = line('XData',IRPathBotMaxDist(1,:),'YData',IRPathBotMaxDist(2,:),...
              'LineStyle',':','MarkerFaceColor', 'blue',...
              'MarkerSize', 1 , 'LineWidth', 2 ,...
              'Marker','.','color','blue');%'LineStyle',':', makes dots
  %2016-9-23F Error Caused by:
  %While setting the 'Marker' property of 'Line': 'circle' is not a valid
value.
  %Use one of these values: '+' | 'o' | '*' | '.' | 'x' | 'square' |
'diamond' | 'v' | '^' | '>' | '<' | 'pentagram' | 'hexagram' | 'none'.
%-----------end IRPathBotMaxDist----------------

if (AN01 == 1);%plot if AN01=1 =ON
%%%%%%%%%%IRPathBotMaxDist;%%%%%%%%%%%%%%%%%%%%%%%%%%%%
hold on;
%-----------IRPathBotMinDist----------------
line05 = line('XData',IRPathBotMinDist(1,:),'YData',IRPathBotMinDist(2,:),...
              'LineStyle','-','MarkerFaceColor', 'red',...
              'MarkerSize', 1 , 'LineWidth', 3 ,...
              'Marker','.','color','red');%'LineStyle',':', makes dots

  %2016-9-26M: An error occurred while running the simulation and the
simulation was terminated
  %Caused by:
  %While setting the 'LineStyle' property of 'Line': '*' is not a valid
value. Use one of these values: '-' | '--' | ':' | '-.' | 'none'.
  %2016-9-23F Error Caused by:
```

```
  %While setting the 'Marker' property of 'Line': 'circle' is not a valid
value.
  %Use one of these values: '+' | 'o' | '*' | '.' | 'x' | 'square' |
'diamond' | 'v' | '^' | '>' | '<' | 'pentagram' | 'hexagram' | 'none'.
%-----------end IRPathBotMaxDist----------------
end;%If (AN01 == 1);%plot if AN01=1 =ON


%Path last to accent over top of IR beam.
hold on;

%First plot line moved here 2016-9-26M
line01 = line('XData',Pos2xPts(1,:),'YData',Pos2xPts(2,:),...
        'MarkerSize', 8 , 'LineWidth', 2 ,...
        'Marker','square','color','black');


hold on;
%-----------Every 10th point----------------
line02 = line('XData',Pos2xPts10th(1,:),'YData',Pos2xPts10th(2,:),...
             'LineStyle','none','MarkerFaceColor', 'black',...
             'Marker','square','color','black');
%-----------end Every 10th point----------------


%{
%add space after % to reactivate block
%--------------------------------------
%annotations as walls and objects in arena
%These must be made to agree manually with ML sub: Niche Layout
%dimensions in fraction of plot area
dimxywhWalls=[ 0.01 0.01 0.99 0.99 ];%dimensions as x,y lower left corner,
width&ht of plot in fraction of area
annotation('rectangle',dimxywhWalls,'FaceColor','cyan',...
           'Color','black','FaceAlpha', 0.2,'LineWidth',2);
%transparency:FaceAlpha from 0 (transparent) to 1 (opaque)
%http://www.mathworks.com/help/matlab/ref/annotation.html?searchHighlight=ann
otate
%annotations in general
%http://www.mathworks.com/help/matlab/ref/annotationrectangle-properties.html
% 'LineWidth',2     0.5 as default

%add objects as rectangles
dimxywhObj1=[ 0.51 0.21 0.04 0.04 ];%dimensions as x,y lower left corner,
width&ht of plot in percent of area
annotation('rectangle',dimxywhObj1,'FaceColor','black',...
           'Color','black','FaceAlpha', 1.0,'LineWidth',2);

dimxywhObj2=[ 0.21 0.51 0.04 0.04 ];%dimensions as x,y lower left corner,
width&ht of plot in percent of area
annotation('rectangle',dimxywhObj2,'FaceColor','black',...
           'Color','black','FaceAlpha', 1.0,'LineWidth',2);

%-----------end annotations----------------
%}
```

```
%add space after % to reactivate block

hold on;
%-----------plotted annotations----------------
%http://www.mathworks.com/help/matlab/ref/annotation.html?searchHighlight=ann
otate
xWalls = [ 0 100 100  0   0 ];%walls
yWalls = [ 0 0   100  100 0 ];%walls
plot(xWalls,yWalls, 'LineWidth', 3, 'Color','black');

%Objects defined in NICHE section

%{
%add space after % on line above to reactivate block
%Start block comment objects 2&1
%Object 1
xObj1 = [ 70 73 73  70 70 ];%Object1 upper right
yObj1 = [ 70 70 73  73 70 ];%Object1 upper right
%xObj1 = [ 50 53 53  50 50 ];%Object1 center
%yObj1 = [ 50 50 53  53 50 ];%Object1 center
%xObj1 = [ 51 54 54  51 51 ];%Object1 original
%yObj1 = [ 21 21 24  24 21 ];%Object1 original
plot(xObj1,yObj1, 'LineWidth', 4, 'Color','black');
%}
%add space after % on line above to reactivate block
%End block comment objects 2&1

%{
%add space after % on line above to reactivate block
%Object 2
xObj2 = [ 17 20 20  17 17 ];%Object2 lower left
yObj2 = [ 20 20 23  23 20 ];%Object2 lower left
%xObj2 = [ 21 24 24  21 21 ];%Object2 original
%yObj2 = [ 51 51 54  54 51 ];%Object2 original
plot(xObj2,yObj2, 'LineWidth', 4, 'Color','black');
%}
%add space after % on line above to reactivate block
%End block comment objects 2&1


%-----------end plotted annotations----------------


%
%str = {'Straight Line Plot','from 1 to 10'};
%strStart = {'START'};
dimxywhStart= [(Pos2xPts(1,1)/100)+.05 (Pos2xPts(2,1)/100)-.15 0.2  0.2];
%annotation('textbox',dim,'String',str,'FitBoxToText','on');

%annotation('textbox',dimxywhStart,...
%           'String','START',...
%           'FitBoxToText','on',...
%           'FontWeight', 'bold', 'Fontsize', 12,...
%           'EdgeColor','none','FaceAlpha', 0.2,'LineWidth',2);
```

```
%transparency:FaceAlpha from 0 (transparent) to 1 (opaque)
%http://www.mathworks.com/help/matlab/ref/annotation.html?searchHighlight=ann
otate
%annotations in general
%http://www.mathworks.com/help/matlab/ref/annotationrectangle-properties.html

%str = {'Straight Line Plot','from 1 to 10'};
%strEnd = {'END'};
dimxywhEnd=[Pos2xPts(1,nPtsMax)/100+.05 Pos2xPts(2,nPtsMax)/100-.15 .3 .3];


%annotation('textbox',dimxywhEnd,'String','END','FitBoxToText','on',...
%           'FontWeight', 'bold', 'Fontsize', 12,...
%           'EdgeColor','none','FaceAlpha', 0.2,'LineWidth',2);
%


refresh(figure1);%redraws the figure identified by h.



%%see above: coder.extrinsic('format','display')
%format bank;%display 2 sig figs to right of decimal
%showabGain3X20=abGain3X20 % print to Command Window
%%disp(abGain3X20) % display in Command Window

 Pos2xPts10th %show every 10th point darker

  %======Print of ANNum value============
  spTotalTime=TotalTime %should agree with StopAtTime (of prog) & StopTime
(of SL)
  %prtdANNum= dANNum %leave semicolon off to print it
  prtANNumTot= ANNumTot %leave semicolon off to print it
  prtANNum= ANNum %leave semicolon off to print it

  %====== END Print of ANNum value============

end;%if (ctime2 > (StopAtTime-0.0001)  )&& (holdtime2<ctime2) && (ctime2 <
StopAtTime+0.0001);
%=========end of Figure inserted here from EXPLORE============================

%======Matrix create examples
%% Creating Multi-Dimensional Arrays
% Multidimensional arrays in MATLAB are created the same way as
% two-dimensional arrays. For example, first define the 3 by 3 matrix, and
% then add a third dimension.

%A = [5 7 8;
%     0 1 9;
%     4 3 6];
%A(:,:,2) = [1 0 4;
%            3 5 6;
%            9 8 7];
```

```
%%
% The CAT function is a useful tool for building multidimensional arrays.
% B = cat(DIM,A1,A2,...) builds a multidimensional array by concatenating
% A1, A2 ... along the dimension DIM.

%B = cat( 3, [2 8; 0 5], [1 3; 7 9], [2 3; 4 6]); %cat along 3rd dimension
% where A1=[2 8; 0 5], A2=[1 3; 7 9], and A3= [2 3; 4 6]

%%
% Calls to CAT can be nested.

%A = cat(3,[9 2; 6 5], [7 1; 8 4]);
%B = cat(3,[3 5; 0 1], [5 6; 2 1]);
%C = cat(4,A,B,cat(3,[1 2; 3 4], [4 3; 2 1]));%cat on 4th dimension

%% Finding the Dimensions
% SIZE and NDIMS return the size and number of dimensions of matrices.

%SzA   = size(A);
%DimsA = ndims(A);
%SzC   = size(C);
%DimsC = ndims(C);

%% Accessing Elements
% To access a single element of a multidimensional array, use integer
% subscripts. For example D(1,2,2,22), using D defined in the previous
% slide, returns 6.
%
% Array subscripts can also be vectors. For example:

%K = C(:,:,1,[1 3]);

%% Selecting 2D Matrices From Multi-Dimensional Arrays
% Functions like EIG that operate on planes or 2D matrices do not accept
% multi-dimensional arrays as arguments. To apply such functions to
% different planes of the multidimensional arrays, use indexing or FOR
% loops. For example:

%A = cat( 3, [1 2 3; 9 8 7; 4 6 5], [0 3 2; 8 8 4; 5 3 5], ...
%                 [6 4 7; 6 8 5; 5 4 3]);
% The EIG function is applied to each of the horizontal 'slices' of A.
%for i = 1:3
%    eig(squeeze(A(i,:,:)))

%======end matrix create examples
%%
theading=heading;
tANNum=ANNum;
uNumOut=uNum;

if (ctime2==0);%output to graph
 EXPLORENumOut=EXPLORENum;
 AVOIDNumOut=AVOIDNum;
 SEEKNumOut=SEEKNum;
else
```

```
 EXPLORENumOut=EXPLORENum/ctime2;
 AVOIDNumOut=AVOIDNum/ctime2;
 SEEKNumOut=SEEKNum/ctime2;
end;%if (ctime2>0);

y = Beh5xPts;
```

ARDUINO PROGRAM

Program for Arduino Nano processor (file: Nano20161102WvPWMSpinAvoidAN01PhD.txt):

```
/*
//===11:14 AM 11/2/2016W  Nano20161102WvPWMSpinAvoidAN01PhD.ino & *.txt
//===1:38 PM 6/8/2015  Nano20150608MvPWMSpin.ino & *.txt
//===2:31 PM 2/21/2015Sat  Nano20150221SatPWMBot.ino & *.txt
//===11:17 AM 12/31/2014W  NanoPWMBot20141231W.ino & *.txt
//===10:28 AM 1/8/2014 Blink20140108.ino & *.txt

//3:59 PM 11/4/2016
//https://www.arduino.cc/en/Guide/Troubleshooting#upload
//had to press reset button during upload to Nano

parts also taken from (on 2:33 PM 2/21/2015):
===7:56 PM 6/18/2014 NanoSDLCD20140618W.ino & *.txt
*/

//

/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
*/

/*
===2:36 PM 2/21/2015
Nano Pin descriptions
Pin assignments:
PWM: 3,5,6,9,10,11
Digital:
D00: RX programming
D01: TX programming
D02: RelayBd1IN1and2D02 (Orange)
D03: PWM MotPWMD03      (Yellow)
D04: RelayBd2IN1and2D04 (Green)
D05: PWM MotPWMD05      (Blue)
D06: open PWM           (Brown) (reserve for: Sonar01CueD06)
D07: open               (White) (reserve for: Sonar01EchoD07)
D08: open AN01          (Orange)
D09: open PWM           (Yellow)
```

D10: open PWM (add: SD card: CS sampling output) (Green)
D11: open PWM (add: SD card: MOSI) (Blue)
D12: open (add: SD card: MISO) (Brown)
D13: open (add: SD card: SCLK) (White)

Analog:
A00/D14: open (add?: Photocell (Green))
A01/D15: open (add?: Thermister(Yellow))
A02/D16: open (IR01A02 (Green)) 1:39 PM 6/27/2015
A03/D17: open (add?: MotA2 (Blue))
A04/D18: open (add?: MotA1 (Brown))
A05/D19: open (add?: MotB2 (White))
A06: open (add: (cannot be digital)(Yellow))
A07: open (add: (cannot be digital)(Green))

http://forum.arduino.cc/index.php?topic=48864.0
Re: Making an analog pin output a signal Response
Jul 17, 2010, 10:35 pm Last Edit: Jul 17, 2010, 10:36 pm by AWOL Reason: 1
Digital pins 14 to 19 are analogue pins 0 to 5.
Just set them up as normal digital pins.
http://arduino.cc/en/Reference/PinMode
 (next line conflicts with above, so try 14 to 19 first)
 The analog input pins can be used as digital pins, referred to as A0, A1, etc.

*/

```
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
//PWM pins 3,5,6,9,10,11
int led = 13;
int RelayBd1IN1and2D02 = 2;
int MotPWMD03 = 3;
int RelayBd2IN1and2D04 = 4;
int MotPWMD05 = 5;
int Sonar01CueD06= 6;
int Sonar01EchoD07= 7;
float Sonar01cm;//distance in cm
int AN01p08=8;//Anticipation cue pin 8

//analog pins
int IR01A02= 2;//Analog A02 chan
float IR01cm= 0;//cm distance for IR01cm on A02
float IRSen01Test= 100;//cm distance scaled for AN01 on
float IRSen01Min= 28;//cm distance min for scaling AN01 on
float IRSen01Max= 50;//50;//cm distance max for scaling AN01 on
float AheadFrac= 1;//scaling of timing for AVOID
```

```
//Movement Cue
int CueMoveStFBSp= 0;//Motion Forw, Back, Spin; 12:56 PM 6/9/2015
  //Cue Move: Stop=0; Forward=1; Back=2; SpinCW=3; SpinCCW=4

//Anticipation: AN01=0(OFF); =1(ON)
int AN01= 0;//Anticipation: AN01=0(OFF); =1(ON) 12:07 PM 11/2/2016

//fan not in use 12:13 PM 6/9/2015
int fanon = 0;


//*****************************************************
//Timers 11:54 AM 6/9/2015; millis() is unsigned long
//12:38 PM 6/9/2015 Binary sketch size: 2,908 bytes (of a 30,720 byte maximum)
//Forward
unsigned long tForwStart=millis();//set Forward start time
unsigned long tForw01=8140;//8140=8.14s//90=35%speed:1.43*2.2s=3140 //2200;//2.2s//2.7 s
default initial
 //tSpinDelay will preempt if a shorter time than tForw01; 2:49 PM 9/28/2015
//tBack
unsigned long tBackStart=millis();//set Backward start time
unsigned long tBack01=3000;//less300=3s//90=35%speed:1.43*2.55s=3650 //2550;//2.55s//3.05
s default initial
//tAVOID
unsigned long tAvoidStart=millis();//set Backward start time
unsigned long tAvoidRun=125;//125=1.25s;//150=1.5s//125=1.25s//200=0.2s//300;//0.3s //0.5s
workds AC test //2.0 s constant SEEK Spin run time
  //140 for Vreg 8AAused set 3.5V on 11:38 AM 3/3/2016//125 OK for batt>5.5V
//Spin/SEEK or AVOID
unsigned long tSpinCW01=1700;//1.7s//1.9 s default initial
unsigned long tSpinCCW01=2000;//2.0 s default initial
unsigned long tSpinDelay=6500;//6.5s//2800;//2.8s//not work right test below:1000;//1.0 s;
constant time to delay in EXPLORE before start SEEK Spin
 //tSpinDelay will preempt if a shorter time than tForw01; 2:49 PM 9/28/2015
unsigned long tSpinRun=2000;//2.0 s constant SEEK Spin run time
unsigned long tSpinWaitStart=millis();//wait Start time in EXPLORE before start SEEK Spin
unsigned long tSpinRunStart=millis();//start time of SEEK Spin
//prints
//CueMoveStFBSp
unsigned long tPrint01Start=millis();
unsigned long tPrint01=3000;
//time tracks
unsigned long tTimeTrak01=0;//
unsigned long tTimeTrak02=0;//
unsigned long tTimeTrak03=0;//
```

```
//*****************************************************
//*****************************************************
void Example()//Template
{
 //Use to create function

}//end: void Example()


//*****************************************************
//*****************************************************
void ANSettings()//Settings for ANticipation
{
 //Note: AN01 digitalRead in SensorsRead();//Read all sensors

 //%%%%%%%%ANticipation additions from MatLab %%%%%%%%%%%%%
 //%set test distance w/ or w/o AN 2016-7-25; 4:30pm.
 if (AN01 > 0)//%AN is ON
 {
  IRSen01Test= IRSen01Max;//%wide range
 }//%1st condition before else end: if (AN01 > 0);//%AN is ON
 else//%AN01=1: AN OFF
 {
  IRSen01Test= IRSen01Min;//%narrow range
 }//end;//%end final: if (AN01 > 0);//%AN is ON

 //Test prints ++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 Serial.print(IRSen01Test);//Serial.print(" , "); Serial.print(Sonar01cm);
 Serial.println(" =IRSen01Test; in ANSettings");



 //==== 2nd set ANticipation additions==========
 //%from AVOID routine in ML; shut off in AVOID: only here now
 //%================================================
 //%AN should *not* be in AVOID; Keep in this Arduino & Cue routine ML Function2
 //%AVOID should just change heading; already cued to turn
 //%================================================
 //%ANticipation: AN01:0=OFF; 1=ON
 if (AN01 > 0)//%ANticipation turned ON
 {
  if (IR01cm>=100)//%equal to ultimate distance
  {
   AheadFrac=1;//%mimimum should never get here, but a protective test
  }//%1st condition before else if end: if (IR01cm=100);//%equal to ultimate distance
  else if (IR01cm<= IRSen01Min+1)
```

```
 {
  AheadFrac=1;//%minimum distance
 }//%2nd condition before else end: if (IR01cm=100);//%equal to ultimate distance
 else//%scaling ratio for AheadFrac
 {
  AheadFrac=1/(IR01cm-IRSen01Min);//%inverse relation, asymptote zero
  //%2016-7-22F ML global variables handle in multiple routines
  //%must also change in Percept function: ML IRSen01fcn(u) and Arduino ChoiceBehavior
  //%ML note: about line 31: IRDistMax=60;%28;%initial value;
  //%ML note: 28cm=11in closeness boundary to cue AVOID
  //%ML note: Also change in AHEAD_AVOID func line 57 as base to divide into

  AheadFrac=min(1,AheadFrac);//%range <==1
 }//end;//%end final: if (IR01cm=100);//%equal to ultimate distance


}//%1st conditon before else end: if (AN01 > 0);//%ANticipation turned ON
else//%ANticipation turned off
{
 AheadFrac=1;//%Max AVOID turn time: tAVOIDRun
}//end;//%end final: if (AN01 > 0);//%ANticipation turned ON

//Test prints ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Serial.print(AheadFrac);//Serial.print(" , "); Serial.print(Sonar01cm);
Serial.println(" =AheadFrac; in ANSettings");

//==== 2nd set ANticipation additions==========


//%%%%%%%END ANticipation additions from MatLab %%%%%%%%%%%%%


}//end: void ANSettings();
//11:00 AM 11/3/2016 Binary sketch size: 8,034 bytes (of a 30,720 byte maximum)


//*****************************************************
//*****************************************************
//Binary sketch size: 6,648 bytes (of a 30,720 byte maximum) 10:55 AM 9/2/2015
void RunBehavior()//Run the cued behavior
{
 //Cue Behavior: WAIT=0; EXPLORE=1; AVOID=2; SEEKCW=3; SEEKCCW=4; Back
only=5
 switch (CueMoveStFBSp)
 {
  case 0:
```

```
 //WAIT
 WaitStop();//WAIT for time<1s
 break;
//----------------------------------------
case 1:
 //EXPLORE
 if ( (millis()-tForwStart) < tForw01)//
 {
  //tSpinDelay will preempt if a shorter time than tForw01; 2:49 PM 9/28/2015
  Forward();
 }//end first if ( (millis()-tForwStart) < tForw01))
 //.............................
 else //no Spin not EXPLOREing: time exceeded
 {
  CueMoveStFBSp= 5;//Back (or SpinCCW) after Forw time ends
  //tAvoidStart=millis();//set Avoid start time
  tBackStart=millis();//Advance of start time to wait before next SEEK Spin
 }//end final if -tForwStart) > tForw01)
 //.............................
 break;
//----------------------------------------
case 2:
 //AVOID
 if ( (millis()-tAvoidStart) < tAvoidRun)
 {
 SpinCCW();//Back and SpinCCW; kept9:55 AM 9/15/2015
 }//end first if ( (millis()-tAvoidStart) < tAvoidRun)
 //.............................
 else //no Spin time exceeded, Done AVOIDing
 {
  tSpinWaitStart= millis();//times start to wit before spin
  CueMoveStFBSp= 1;//Forw after AVOID/Spin time ends
  tForwStart=millis();//Advance of start time to wait before next Forw
 }//end final if-tAvoidStart) < tAvoidRun)
 //.............................
 break;
//----------------------------------------
case 3:
 //SEEK SpinCW
 if ( (millis()-tSpinRunStart) < tSpinRun)
 {
 SpinCW();//Back and SpinCW
 }//end first if ( (millis()-tSpinRunStart) < tSpinRun)
 //.............................
 else //no Spin time exceeded, not EXPLOREing
 {
```

```
  tSpinWaitStart= millis();//times start to wait before spin
  CueMoveStFBSp= 1;//Forw after SEEK/Spin time ends
  tForwStart=millis();//Advance of start time to wait before next Forw
 }//end final if-tSpinRunStart) < tSpinRun)
 //.....................................
 break;
//-----------------------------------------
case 4:
 //SEEK BackSpinCCW; 9:52 AM 9/15/2015
 //Previous: SEEK SpinCCW; 9:52 AM 9/15/2015
 if ( (millis()-tSpinRunStart) < tSpinRun)
 {
 //BackSpinCCW();//Back and SpinCCW;9:53 AM 9/15/2015
 SpinCCW();//Back and SpinCCW; revert 11:25 AM 9/15/2015
 }//end first if ( (millis()-tSpinRunStart) < tSpinRun)
 //.....................................
 else //no Spin time exceeded, not EXPLOREing
 {
  tSpinWaitStart= millis();//times start to wait before spin
  CueMoveStFBSp= 1;//Forw after SEEK/Spin time ends
  tForwStart=millis();//Advance of start time to wait before next Forw
 }//end final if-tSpinRunStart) < tSpinRun)
 //.....................................
 break;
//-----------------------------------------
case 5:
 //Back: straight Back
 if ( (millis()-tBackStart) < tBack01)//
 {
 Back();//Back straight
 }//end first if ( (millis()-tBackStart) < tBack01))
 //.....................................
 else //no Spin, not Backing: time exceeded
 {
  tSpinWaitStart= millis();//times start to wit before spin
  tForwStart=millis();//Advance of start time to wait before next Forw
  CueMoveStFBSp= 1;//Forw after Back time ends
  tBackStart=millis();//Advance of start time to wait before next SEEK Spin
 }//end final if -tBackStart) > tBack01)
 //.....................................
 break;
//-----------------------------------------
default:
 // if nothing else matches, do the default
 // default is optional
 break;
```

```
 }//end switch (CueMoveStFBSp)
}//end: void RunBehavior()


//*****************************************************
//*****************************************************
//Binary sketch size: 6,318 bytes (of a 30,720 byte maximum) 3:34 PM 9/1/2015
void ChoiceBehavior()
{
 //Use to create function 6:10 PM 8/31/2015
 //Cue Behavior: WAIT=0; EXPLORE=1; AVOID=2; SEEKCW=3; SEEKCCW=4; Back
only=5
 //Cue OLD Move: Stop=0; Forward=1; Back=2; SpinCW=3; SpinCCW=4
 //.............................
 if (millis() < 1000)//1 sec
 {//WAIT before start
  CueMoveStFBSp= 0;//Motion Forw, Back, Spin;
  //Cue Behavior: WAIT=0; EXPLORE=1; AVOID=2; SEEKCW=3; SEEKCCW=4
  //
  //Test prints ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  Serial.print(CueMoveStFBSp);//Serial.print(" , "); Serial.print(Sonar01cm);
  Serial.println(" =CueMoveStFBSp; millis() < 1000");
  //
 }//end first test if (millis() < 1000)
 else
 {
  if (CueMoveStFBSp == 0)
  {
   tSpinWaitStart= millis();//times start to wait before spin
   tForwStart=millis();
   CueMoveStFBSp= 1;
  }//EXPLORE start
 }//end final if (millis() < 1000)



 //%=============AVOID Start Cue================================
 //%AVOID only if AVOID started (CueMoveStFBSp== 4) or ==3 in ML
 //%AVOID no longer interrupts SEEK
 // || is OR operator
 if(CueMoveStFBSp== 1 || CueMoveStFBSp== 2)//%(Cue EXPLORE=1 || Cue AVOID=2)
 {


 //.............................
 //Next line changed to IRSen01Test from 28 on 2016-11-3R, 2pm
```

```
 if (IR01cm < IRSen01Test)//%28)//28cm=11in//25 cm=10in
 //IRSen01Test is either IRSen01Min(=28) or IRSen01max(=50) in AN01Settings 2:14 PM
11/3/2016
 {//AVOID flag: Back & turn from wall

  if (CueMoveStFBSp != 2)//not equal to 2
  {
   tAvoidStart=millis();
  }//AVOID start
  CueMoveStFBSp= 2;//AVOID start;
  //Cue Behavior: WAIT=0; EXPLORE=1; AVOID=2; SEEKCW=3; SEEKCCW=4

  //.....................
  if ( ((millis()-tAvoidStart) > tAvoidRun*AheadFrac) && (CueMoveStFBSp == 2)) //2 sec; &&
is AND
  //%if ( ((millis()-tAvoidStart) > tAvoidRun) && (CueMoveStFBSp == 2)) //2 sec; && is AND
  {
   tSpinWaitStart= millis();//times start to wait before spin
   tForwStart=millis();
   CueMoveStFBSp= 1;//EXPLORE start;
   //Cue Behavior: WAIT=0; EXPLORE=1; AVOID=2; SEEKCW=3; SEEKCCW=4
  }//end if ( (millis()-tAvoidStart) > tAvoidRun)
  //
  //Test prints ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  Serial.print(CueMoveStFBSp);//Serial.print(" , "); Serial.print(Sonar01cm);
  Serial.println(" =CueMoveStFBSp; IR01cm < 25");
  //
 }//end first test if (IR01cm <IRSen01Test=28or50)
 else //for this section, IR01cm>IRSen01Test=28or50, so not near wall
 {
  if ( ((millis()-tAvoidStart) > tAvoidRun*AheadFrac) && (CueMoveStFBSp == 2)) //2 sec; &&
is AND
  //%if ( ((millis()-tAvoidStart) > tAvoidRun) && (CueMoveStFBSp == 2)) //2 sec; && is AND
  {
   tSpinWaitStart= millis();//times start to wit before spin
   tForwStart=millis();
   CueMoveStFBSp= 1;//EXPLORE start; a
   //Cue Behavior: WAIT=0; EXPLORE=1; AVOID=2; SEEKCW=3; SEEKCCW=4
  }//end if ( (millis()-tAvoidStart) > tAvoidRun)
  //
  //Test prints ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  Serial.print(CueMoveStFBSp);//Serial.print(" , "); Serial.print(Sonar01cm);
  Serial.println(" =CueMoveStFBSp; IR01cm > 25");
  //
  //.............................
 }//end final if (IR01cm < 25)
```

```
//.............................
if (Sonar01cm < 20)//20cm
{//ANticipate: AVOID flag: Back & turn from wall
 //need rule here
}//end if (Sonar01cm < 20)


}//%end if(CueMoveStFBSp== 1 || CueMoveStFBSp== 2)//%(Cue EXPLORE || Cue AVOID);
// || is OR operator
//%============END: AVOID Start Cue================================



//.............................
if (CueMoveStFBSp== 1)//SEEK Spin start and test to continue/end
{//EXPLORING timed for Spin of SEEK
//
 //Test prints
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 Serial.print(CueMoveStFBSp);Serial.print(" , "); Serial.print(millis()-tSpinWaitStart);
 Serial.println(" =CueMoveStFBSp; millis()-tSpinWaitStart; CueMoveStFBSp== 1");
 //
  if ( (millis()-tSpinWaitStart) > tSpinDelay)//1 sec
  {
   tSpinWaitStart=millis();
   tSpinRunStart=millis();
   CueMoveStFBSp= 4;//Motion Spin;
   //Cue Behavior: WAIT=0; EXPLORE=1; AVOID=2; SEEKCW=3; SEEKCCW=4
  }//end first if ( (millis()-tSpinWaitStart> tSpinDelay)
  //.............................
 else //no Spin not EXPLOREing
 {
  CueMoveStFBSp= CueMoveStFBSp;//no change to CueMoveStFBSp
  //junk//let time pass by no update of start time: tSpinRunStart
  //junk//tSpinWaitStart=millis();//Advance of start time to wait before next SEEK Spin
 }//end final if ( (millis()-tSpinWaitStart) > tSpinDelay)
 //*****************************************

}//end first test before else if: if (CueMoveStFBSp== 1)


//%.............................
//%...SEEK RUNS HERE ..... SEEKRUNSHERE .....elseif,else,end}
//%.............................
else if (CueMoveStFBSp== 4)//SEEK Spin in Progress //% ==3 in SLML
{
```

```
 if ( (millis()-tSpinRunStart) > tSpinRun)//2.0 sec
 {
  tSpinWaitStart= millis();//times start to wait before spin
  tSpinRunStart=millis();
  tForwStart=millis();
  CueMoveStFBSp= 1;//EXPLORE again
  //Cue Behavior: WAIT=0; EXPLORE=1; AVOID=2; SEEKCW=3; SEEKCCW=4
 //
 //Test prints ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 Serial.print(CueMoveStFBSp);//Serial.print(" , "); Serial.print(Sonar01cm);
 Serial.println(" =CueMoveStFBSp; CueMoveStFBSp== 4");
 //


 }//end 1st condition before else: if ( (millis()-tSpinRunStart) > tSpinRun)//2.0 sec

 else////SEEK still spinning
 {

  //%---check if AN=1 is ON (decode to Arduino 4:23 PM 11/3/2016)-----
  if (AN01 > 0)//%AN is ON
  {
   //.....................................
   //%set above: IRSen01Test= IRSen01Max;%wide range
   if (IR01cm < IRSen01Max)//%object in range of IRSen01Max
   {
    //%only if object in range of IRSen01Max
    tSpinWaitStart= millis();//times start to wait before spin
    tSpinRunStart=millis();
    tForwStart=millis();
    CueMoveStFBSp= 1;//EXPLORE again
    //Cue Behavior: WAIT=0; EXPLORE=1; AVOID=2; SEEKCW=3; SEEKCCW=4
   }//1st condition before else end: if (IR01cm < IRSen01Max)//%object in range of
IRSen01Max

   else//Object outside range of IRSen01Max
   {
    //%no change if no object in range of IRSen01Max
   }//end final: if (IR01cm < IRSen01Max)//%object in range of IRSen01Max
   //.....................................

  }//1st condition end:if (AN01 > 0)//%AN is ON
  else//AN01=0: AN OFF
  {
   //no change if AN is not ON
  }//end final: if (AN01 > 0)//%AN is ON
```

```
  //%---end check if AN=1 is ON (decode to Arduino 4:23 PM 11/3/2016)-----

 //Binary sketch size: 8,270 bytes (of a 30,720 byte maximum) 4:49 PM 11/3/2016



 }//end final: if (millis()-tSpinRunStart) > tSpinRun)//2.0 sec
}//end second before else: if (CueMoveStFBSp== 4)

//.....................
else
{
 CueMoveStFBSp= CueMoveStFBSp;//no change to CueMoveStFBSp
}//end final if (CueMoveStFBSp== 1)

//
if ( (millis()-tPrint01Start) > tPrint01)
{
 tTimeTrak01= millis()-tPrint01Start;//time gap
 tPrint01Start=millis();//reset time for printing
 Serial.print(CueMoveStFBSp);Serial.print(" , "); Serial.print(tTimeTrak01);
 Serial.println(" =CueMoveStFBSp, tTimeTrak01, time=tPrint01");
}//end first if ( (millis()-tPrint01Start) > tPrint01)
//
}//end: ChoiceBehavior()


//*******************************************************
//*******************************************************
void ScreenPrint()
{
 //Use to create function
  //
  Serial.print(IR01cm); Serial.print(" , ");Serial.print(Sonar01cm);
  Serial.println(" =IR01cm,Sonar01cm");
  //

}//end: void ScreenPrint();

//*******************************************************
//*******************************************************
float Sonarcm (int CuePin, int EchoPin)//12:50 PM 6/27/2015
{
 //Cue and read a sonar
 //4 pin sonar: HC-SR04
 //http://www.grook.net/how-to-make-radar-using-arduino-uno
```

```
  digitalWrite(CuePin, LOW);
  delayMicroseconds(2);
  digitalWrite(CuePin, HIGH);
  delayMicroseconds(10);
  digitalWrite(CuePin, LOW);
  // Distance Calculation
  float distance = pulseIn(EchoPin, HIGH,5800);//58*100cmmax=5800 timeout
  distance= distance/58;//distance in cm= time/58 conversion factor
  //float distance=0;//test value only

  return(distance);
}//end: float Sonarcm (CuePin,EchoPin)//12:50 PM 6/27/2015

//*******************************************************
//*******************************************************

void SensorsRead()//12:27 PM 6/27/2015
{
 //Read as variable (1st) & store in an array (2nd)
  //Anticiption: AN01=0(OFF; =1(ON)
  AN01=digitalRead(AN01p08);
  //Test prints ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  //Serial.print(AN01);//Serial.print(" , "); Serial.print(Sonar01cm);
  //Serial.println(" =AN01; in SensorsRead");

  //IR01cm= nearby distance on A02
  //Sonar01cm= far distance using Cue D06 and Echo read D07

 //IR01 read distance nearby
 //Noah Stahl 5/25/2011 http://arduinomega.blogspot.com Arduino Mega 2560
 //http://arduinomega.blogspot.com/2011/05/infrared-long-range-sensor-gift-of.html
 //Calibration in Excel file: RobotArch20150122RCausalDiagStocks&Flows.xls :
ANIMIRsensor20150508F
 /*===2:22 PM 6/27/2015
 See file for calibration:
 RobotArch20150122RCausalDiagStocks&Flows.xls : ANIMIRsensor20150508F
 IR sensor:
 GP2Y0A02YK0F, 20 to 150cm
 For V=< 1.15:
 Inverse Graph: longer distance >50 cm
 y=52/V + 5
 For V>1.15:
 Direct Graph: close distance <50 cm
 y=-18.6V+62.4
 */
 //IR01cm must be a float or double
```

```
 IR01cm= analogRead(IR01A02);//Initial reading
 IR01cm= IR01cm*5/1024;//Volts: calc must be on different line than analogRead()
 //2:20 PM 9/2/2015
 //analogRead(): 5 volts / 1024 units, so multiply reading by *5/1024
 if (IR01cm < 1.16) //long distance > 50 cm
  {
   IR01cm= 52/IR01cm + 5;//conversion to cm
  }
  else  //short distance < 50 cm
  {
   IR01cm= -18.6*IR01cm + 62.4;//conversion to cm
  }//end: if (IR01cm < 1.16)
 //alternative default from:
 //http://arduinomega.blogspot.com/2011/05/infrared-long-range-sensor-gift-of.html
 //IR01cm=   10650.08 * pow( IR01cm, -0.935) - 10;
 //inches= 4192.936 * pow( IR01cm, -0.935) - 3.937;

 //https://www.arduino.cc/en/Serial/Print
 //Floats are similarly printed as ASCII digits, defaulting to two decimal places.
 //Test prints ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 Serial.print(IR01cm);//Serial.print(" , "); Serial.print(Sonar01cm);
 Serial.println(" =IR01cm; in SensorsRead");

 //*****************************
 //Sonar01 read distance far away
 //4 pin sonar: HC-SR04
 //http://www.grook.net/how-to-make-radar-using-arduino-uno
 //int Sonar01CueD06= 6; //int Sonar01EchoD07= 7;
 Sonar01cm= Sonarcm(Sonar01CueD06,Sonar01EchoD07);


}//end: void SensorsRead();//12:27 PM 6/27/2015
//Binary sketch size: 4,660 bytes (of a 30,720 byte maximum) 4:26 PM 6/27/2015

//*********************************************************
//*********************************************************
void WaitStop() //8:57 AM 9/2/2015
{
  //WaitStop: HIGH leaves relay at default Forward
  //HIGH is unswitched relay (NC), LOW switches relay from NC to NO
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  digitalWrite(RelayBd1IN1and2D02, HIGH);//RelayIN1on (HIGH=OFF=NC=Forward;
LOW=ON=NO=Backward )
  analogWrite(MotPWMD03, 0);//Stop speed Forward; Min=0; Max=255
  digitalWrite(RelayBd2IN1and2D04, LOW);//RelayIN2off (HIGH=OFF=NC=Forward;
LOW=ON=NO=Backward )
```

```
  analogWrite(MotPWMD05, 0);//Stop speed; Min=0; Max=255
  //delay(tSpinCW01);//=1900;//1.9 s; not needed: test in ChoiceBehavior()

}//end: WaitStop() //8:57 AM 9/2/2015

//*********************************************************
//*********************************************************
void SpinCW() //12:18 PM 6/9/2015 CW only //1:41 PM 6/8/2015
{
  //SpinCW: HIGH leaves relay at default Forward
  //HIGH is unswitched relay (NC), LOW switches relay from NC to NO
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  digitalWrite(RelayBd1IN1and2D02, HIGH);//RelayIN1on (HIGH=OFF=NC=Forward;
LOW=ON=NO=Backward )
  analogWrite(MotPWMD03, 200);//Full speed Forward; Max=255
  digitalWrite(RelayBd2IN1and2D04, LOW);//RelayIN2off (HIGH=OFF=NC=Forward;
LOW=ON=NO=Backward )
  analogWrite(MotPWMD05, 200);//Full speed; Max=255
  //delay(tSpinCW01);//=1900;//1.9 s; timing in RunBehavior()
  //delay(1900);             // wait for 2.#s (about)
  //delay(4000);             // wait for 4.0s

}//end: void SpinCW() //12:18 PM 6/9/2015 CW only //1:41 PM 6/8/2015

//*********************************************************
//*********************************************************
void BackSpinCCW() //9:45 AM 9/15/2015 //12:20 PM 6/9/2015 CCW only //1:41 PM 6/8/2015
{
  //9:43 AM 9/15/2015 obsolete for SEEK SpinCCW(), replace: BackSpinCCW()
  //9:56 AM 9/15/2015 kept SpinCCW() for AVOID
  //10:45 AM 9/15/2015 Low batt<5.5V does not SpinCCW well; new batts my be OK
  //BackSpinCCW: HIGH leaves relay at default Forward
  //HIGH is unswitched relay (NC), LOW switches relay from NC to NO
  digitalWrite(led, LOW);   // turn the LED OFF by making the voltage LOW
  digitalWrite(RelayBd1IN1and2D02, LOW);//RelayIN1off (HIGH=OFF=NC=Forward;
LOW=ON=NO=Backward )
  analogWrite(MotPWMD03, 150);//200);//near Full speed; Max=255
  digitalWrite(RelayBd2IN1and2D04, LOW);//Prev: HIGH);//RelayIN2on
(HIGH=OFF=NC=Forward; LOW=ON=NO=Backward )
  analogWrite(MotPWMD05, 25);//Prev: 200);//Full speed; Max=255
  //delay(tSpinCCW01);//=2000;//2.0 s; timing in RunBehavior()
  //delay(2000);             // wait for 1.#s
  //delay(4000);             // wait for 4.0s
```

```
}//end: void BackSpinCCW() //9:45 AM 9/15/2015//12:20 PM 6/9/2015 CCW only //1:41 PM
6/8/2015

//*********************************************************
//*********************************************************
void SpinCCW() //12:20 PM 6/9/2015 CCW only //1:41 PM 6/8/2015
{
  //9:43 AM 9/15/2015 obsolete for SEEK SpinCCW(), replace: BackSpinCCW()
  //9:56 AM 9/15/2015 kept SpinCCW() for AVOID
  //SpinCCW: HIGH leaves relay at default Forward
  //HIGH is unswitched relay (NC), LOW switches relay from NC to NO
  digitalWrite(led, LOW);   // turn the LED OFF by making the voltage LOW
  digitalWrite(RelayBd1IN1and2D02, LOW);//RelayIN1off (HIGH=OFF=NC=Forward;
LOW=ON=NO=Backward )
  analogWrite(MotPWMD03, 200);//prev200);//Full speed Forward; Max=255
  digitalWrite(RelayBd2IN1and2D04, HIGH);//RelayIN2on (HIGH=OFF=NC=Forward;
LOW=ON=NO=Backward )
  analogWrite(MotPWMD05, 200);//prev200);//Full speed; Max=255
  //delay(tSpinCCW01);//=2000;//2.0 s; timing in RunBehavior()
  //delay(2000);            // wait for 1.#s
  //delay(4000);            // wait for 4.0s


}//end: void SpinCCW() //12:20 PM 6/9/2015 CCW only //1:41 PM 6/8/2015


//*********************************************************
//*********************************************************
void Forward() //1:08 PM 6/9/2015
{

  //Forward: HIGH leaves relay at default Forward
  //HIGH is unswitched relay (NC), LOW switches relay from NC to NO
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  digitalWrite(RelayBd1IN1and2D02, HIGH);//RelayIN1off (HIGH=OFF=NC=Forward;
LOW=ON=NO=Backward )
  analogWrite(MotPWMD03, 90);//90=35% speed//125);//Half speed Forward; Max=255
  //digitalWrite(MotPWMD03, HIGH);   // turn the LED on (HIGH is the voltage level)
  digitalWrite(RelayBd2IN1and2D04, HIGH);//RelayIN2off (HIGH=OFF=NC=Forward;
LOW=ON=NO=Backward )
  analogWrite(MotPWMD05, 90);//90=35% speed//125);//Half speed; Max=255
  //digitalWrite(MotPWMD05, HIGH);   // turn the LED on (HIGH is the voltage level)
  //delay(tForw01);//=2700;//2.7 s; timing in RunBehavior()
  //delay(2700);            // wait for 2.#s
  //delay(4000);            // wait for 4.0s
```

```
}//end: Forward() //1:08 PM 6/9/2015

//*********************************************************
//*********************************************************
void Back() //1:09 PM 6/9/2015
{
  //Back: LOW switches relay to go Backward
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  digitalWrite(RelayBd1IN1and2D02, LOW);//RelayBd1IN1and2D02 on
(LOW=ON=NO=Backward)
  analogWrite(MotPWMD03, 90);//90=35% speed//125);//Half speed; Max=255
  //digitalWrite(MotPWMD03, HIGH);   // turn the LED on (HIGH is the voltage level)
  digitalWrite(RelayBd2IN1and2D04, LOW);//RelayBd2IN1and2D04 on
(LOW=ON=NO=Backward)
  analogWrite(MotPWMD05, 90);//90=35% speed//125);//Half speed; Max=255
  //digitalWrite(MotPWMD05, HIGH);   // turn the LED on (HIGH is the voltage level)
  //delay(tBack01);//=3050;//3.05 s; timing in RunBehavior()
  //delay(3000);              // wait for 3.#s
  //delay(4000);              // wait for 4.0s

}//end: Back() //1:09 PM 6/9/2015
//1:14 PM 6/9/2015;Binary sketch size: 2,942 bytes (of a 30,720 byte maximum)


//*********************************************************
//*********************************************************


// the setup routine runs once when you press reset:
void setup()
{
// initialize serial communications at 9600 bps:
 Serial.begin(9600);
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
  pinMode(RelayBd1IN1and2D02, OUTPUT);
  pinMode(MotPWMD03, OUTPUT);
  pinMode(RelayBd2IN1and2D04, OUTPUT);
  pinMode(MotPWMD05, OUTPUT);
  digitalWrite(RelayBd1IN1and2D02, HIGH);//RelayIN1off (HIGH=OFF=NC, LOW=ON=NO )
  digitalWrite(MotPWMD03, LOW);   // turn the LED on (HIGH is the voltage level)
  digitalWrite(RelayBd2IN1and2D04, HIGH);//RelayIN2off (HIGH=OFF=NC, LOW=ON=NO )
  pinMode(led05, OUTPUT); //fan High
  digitalWrite(led04, LOW);   // turn the LED on (HIGH is the voltage level)
  digitalWrite(MotPWMD05, LOW);   // turn the LED on (HIGH is the voltage level)
```

```
  //Sonar01cm pin directions
  pinMode(Sonar01CueD06, OUTPUT);
  pinMode(Sonar01EchoD07, INPUT);

  //Anticipation pin direction
  pinMode(AN01p08, INPUT);

}//end: void setup()

//*********************************************************
//*********************************************************

// the loop routine runs over and over again forever:
void loop()
{
 //Thermistor();//F from A01

 SensorsRead();//Read all sensors
 ScreenPrint();//Print to computer screen sensor readings

 ANSettings();//Settings for ANticipation
 ChoiceBehavior();//Choice of behavior
 RunBehavior();//Run the cued behavior

 //test back with delay
 //Back();delay(3000);//Back only; base


/* old code before 2:10 PM 9/2/2015
 //Movement cues with delays
 Forward();//Forward only; base
 Back();//Back only; base
 Forward();//Forward only; base
 Back();//Back only; base
 //ForwardBack();//Revised and gone 1:05 PM 6/9/2015; Move routine, simple
 SpinCW();//Spin CW only 12:27 PM 6/9/2015
 SpinCCW();//Spin CCW only 12:28 PM 6/9/2015
 //Spin();//revised and gone 12:27 PM 6/9/2015 //Spin both CW & CCW 2:08 PM 6/8/2015
*/ //end old code before 2:10 PM 9/2/2015


  for (int irun=1; irun < 11; irun++)
  {//
  //

  //Fan control (Extra routine)
```

```
  //for (int irun=1; irun < 11; irun++)
  //{
   if (irun == 10)
   { //n cycles pass
    if (fanon == 0)
    {//turn fan on
     fanon = 1;
     digitalWrite(led05, HIGH);   // turn the LED on (HIGH is the voltage level)
     //delay(4000);               // wait for a second
    }
     else
     {//turn fan off
      fanon = 0;
      digitalWrite(led05, LOW);   // turn the LED on (HIGH is the voltage level)
      //delay(100);               // wait for a 0.1 s
    }//end: if (fanon = 0)
   //
   }
    else
   {// do nothing until n cycles
   }//end: if (irun == 20)

   }// end: for (int irun=1; irun = 21; irun++)


}//end: void loop()


//eof
```

A4. ANTICIPATION METRIC

A specific metric (ANNum) was developed for operation of the TOURIST robot considering the distance measured by the IR beam and the area covered by the motion of the robot at any time instance (explained in this Appendix A4). The metric (ANNum) can be used to compare travel of robot paths from ANSIM model for a set time and arena configuration. The same method is used for EXPLORE and AVOID behaviors both with no anticipation (NO AN) and with anticipation on (AN ON). However, for SEEK there is no calculation with NO AN since the spin during SEEK in that instance is not interrupted. However, with AN ON the area is calculated during the SEEK behvavior because the IR sensor is actively perceiving the niche during the spin, and the spin is interrupted when an object or wall is perceived.

ANNum is integrated over the entire travel time, normalized by the total time to allow comparison among paths of different time duration:

$$dANNum(t) = D_{IR} * \text{DistanceTraveled} \quad \text{(A4.1)}$$

$$ANNumTot(t) = ANNumTot(t-1) + dANNum(t) \quad \text{(A4.2)}$$

$$ANNum = ANNumTot(t) / TotalTime \quad \text{(A4.3)}$$

where each of the terms are:

D_{IR}: IR beam distance as the minimum of beam extension used (for NO AN: 28 cm; for AN ON: 50 cm.) or the distance reading to a wall or object [D_{IR} = min (IRMaxDist, IRReading).]

DistanceTraveled: forward distance in EXPLORE (a parallelogram), or the heading change in either AVOID or SEEK behaviors as a distance at the extreme of the beam (a triangle, so must divide by 2 for the area covered).

dANNum(t): calculated change at any time interval.

ANNumTot(t): total integrated value over the path at time, t.

ANNum: value at any time as normalized for total time of travel.

The result of A4.2 is to integrate over time, and A4.3 normalizes for the total travel time.

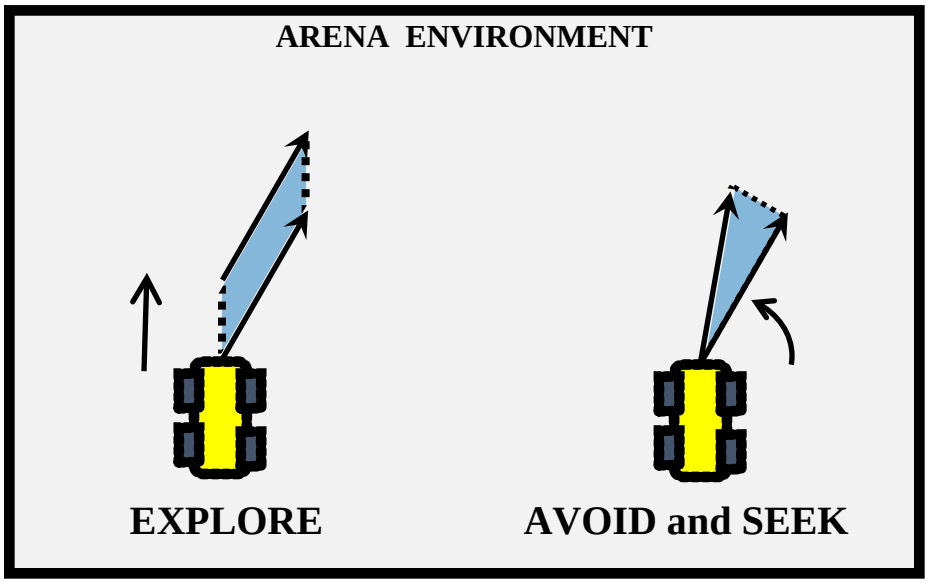


Figure 72. Metric ANNum is the integral of incremental area coverage of the niche environment, shown for EXPLORE (left) as a parallelogram, and both AVOID and SEEK (right) as a triangle.

There are potential relative sources of error in this calculation due to geometry (Fig. 72). For EXPLORE, due to the angle of 30 degrees offset from forward for the IR sensor beam, the parallelogram has a height that is ht_p= sin 30 * DistanceTraveled = 0.5 * DistanceTraveled, so adjustment by the factor of 0.5 must be made. For AVOID and SEEK, since the area of a triangle is A_t= 0.5 bh, the factor of 0.5 must be included. This uses the base as b= IR beam length, and the distance moved at the tip of the beam to be h= height of the triangle. For small angles it should be sufficient to consider a right angle between the IR beam and the tip distance traveled. As the SEEK behavior may spin as much as 270 degrees, the summation of all the triangles made

during the spin over multiple time increments accounts for the total covered area. Integration of these areas over both time and motion calculates the relative area covered that is able to be compared for the paths traveled over time, and normalized for total time traveled.

Comparisons for two basic arena configurations show that the ANNum value is greater with anticipation on (AN ON) as compared with no anticipation (NO AN) (Table A.4.1, corresponding runs are shown in Figs. 54 and 66 in the main text).

Table A.4.1. ANNum values with no anticipation and with anticipation on.

	No Anticipation (cm^2/s)	Anticipation On (cm^2/s)
Walls Only	115	274
With 2 Objects	118	383

ANNum with Walls Only is about 2.38 times greater with AN ON. Similarly, With 2 Objects in the arena the ANNum value is greater by about 3.25 times. Since the ANNum value is essentially a normalized cummulative area calculation, it can be said the robot perceives and responds to an area 2 to 3 times greater with anticipation on (AN ON). Comparisons over time for ANNum calculations show that values are initially high for the area considered, but ongoing the normalization factor of time reduced the value to a rather stable one (Figs. 73 and 74). Therefore, overall the metric ANNum can be used to compare various responses of the robot to layouts of an arena. Generally, anticipation allows for higher values to occur, reflecting more behavior response per unit time, while the metric stabilizes to an asymptotic value as normalized over increasing time.

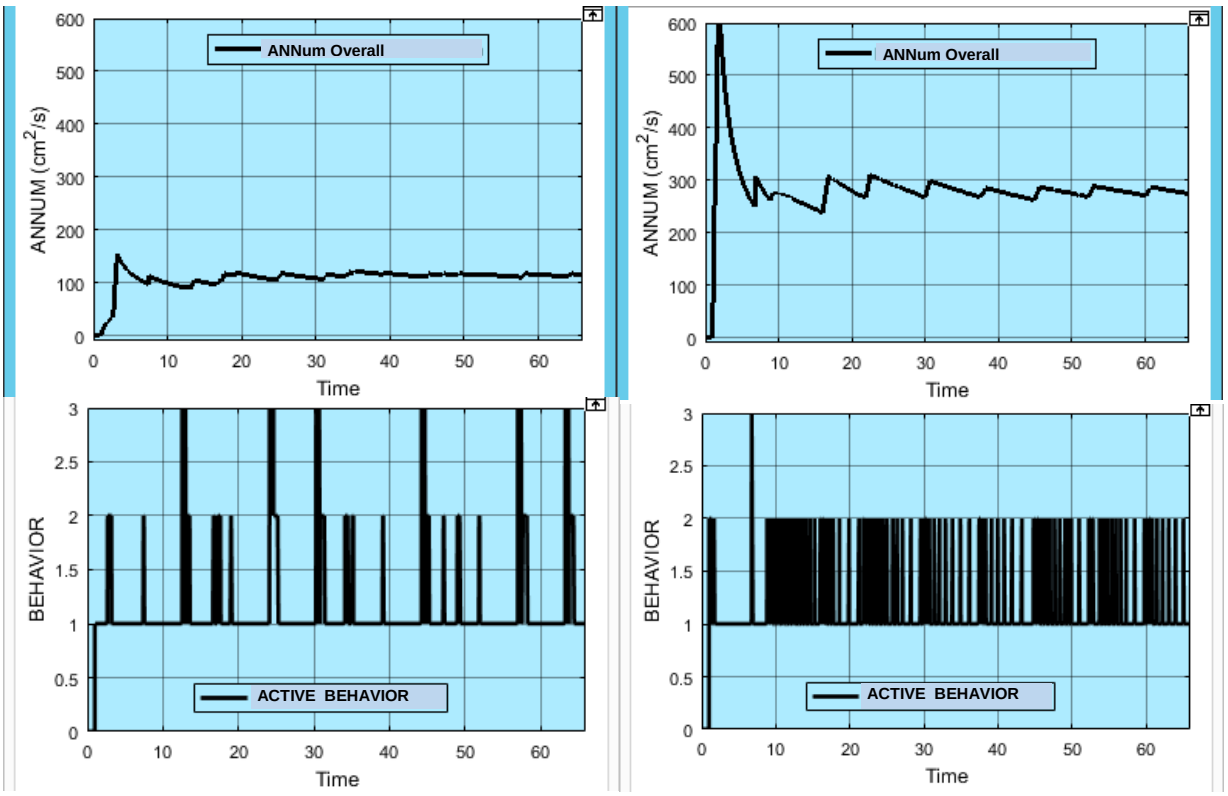
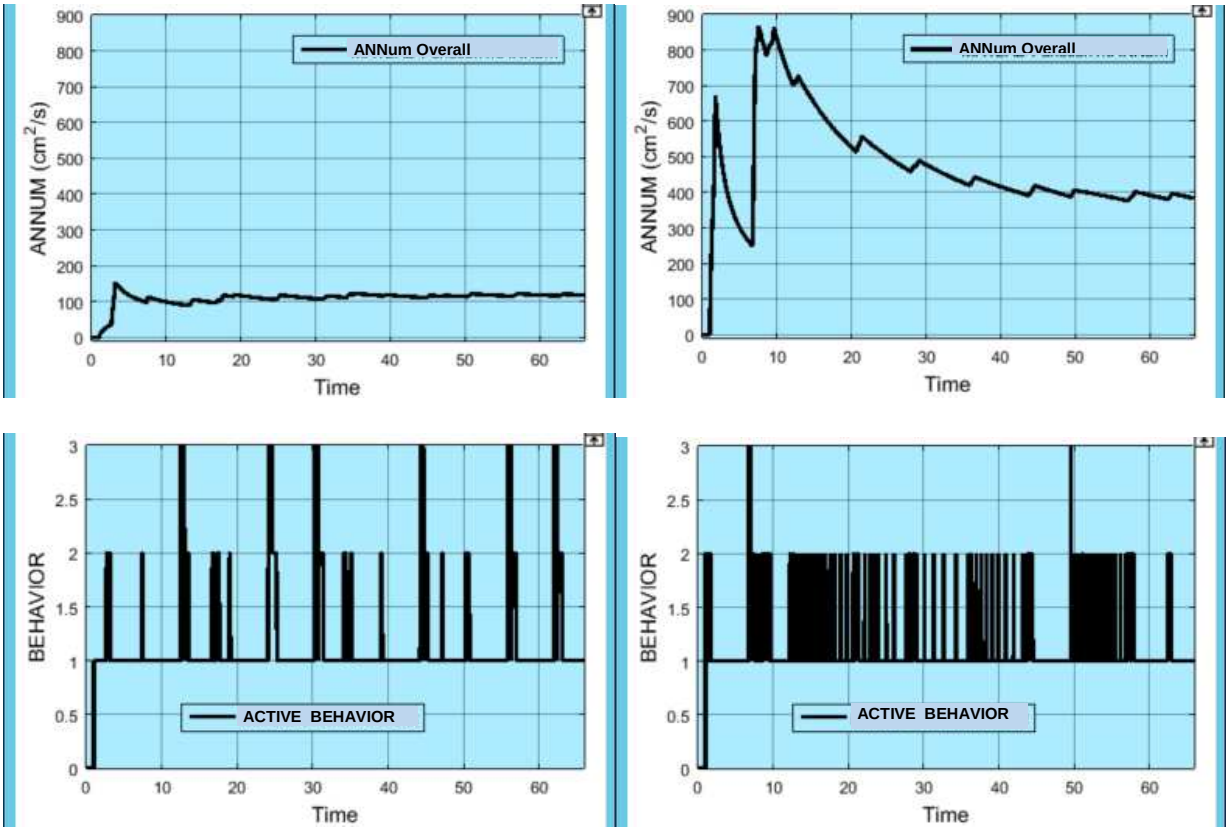


Figure 73. TOURIST path simulated for 66 s (21*pi) and Walls Only. ANNum metric over time for NO AN, ANNum= 115 cm^2/s (left) and AN ON, ANNum= 274 cm^2/s (right). Behaviors shown (bottom) to compare with ANNum calculated (above).



NO ANTICIPATION. 28 cm **ANTICIPATION ON. 28 & 50 cm**

Figure 74. TOURIST path simulated for 66 s (21*pi) and With 2 Objects. ANNum metric over time for NO AN, ANNum= 118 cm^2/s (left) and AN ON, ANNum= 383 cm^2/s (right). Behaviors shown (bottom) to compare with ANNum calculated (above).

A5. ADDITIONAL REFERENCES (UNCITED)

Design and System Dynamics

Ali, M. (1994). Exploration-based design synthesis of Behavior-based autonomous robots. Ph.D Dissertation. Colorado State University. 130 pp.

Amir, E. & Maynard-Zhang, P. (2004). Logic-based subsumption architecture. *Artificial Intelligence*. 153, 167-237.

Brooks, R. (1996). Behavior-Based Humanoid Robotics. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robotics and Systems:IROS 96*. Osaka, Japan.

Brooks, R. (1997). From earwigs to humans. *Robotics and Automous Systems*, 20, 291-304.

Brooks, R., Breazeal, C., Marjanovic, M., Scassellati, B., & Williamson, M., (1998). The Cog project; building a humanoid robot. in C. Nehaniv, ed., *Computation for Metaphors, Analogy and Agents*, Vol. 1562 of Springer Lecture Notes in Artificial Intelligence, Springer-Verlag, 1998. Accesed June 1, 2015 http://groups.csail.mit.edu/lbr/hrg/1998/springer-final-cog.pdf and listed http://groups.csail.mit.edu/lbr/humanoid-robotics-group/cog/publications.html.

Cherian, S. (1995). Designing behavior networks for autonomous robotic systems. PhD Dissertation. Colorado State University. 106 pp.

Hoffmann, M., Schmidt, N., Pfeifer, R., Engel, A., & Maye, A. (2012). Using Sensorimotor Contingencies for Terrain Discrimination and Adaptive Walking Behavior in the Quadruped Robot Puppy. In: From Animals to Animats 12, Lecture notes in Computer Science, 7426, 54-64. Accessed September 29, 2014: http://link.springer.com/chapter/10.1007/978-3-642-33093-3_6.

Kaiser, C. (2009). Interaction Space abstractions: Design methodologies and tools for autonomous robot design and modeling. Ph.D. Dissertation. Colorado State University. 189 pp.

Laszlo, E. (1972). The relevance of General Systems Theory: Papers presented to Ludwig von Bertalanffy on his seventieth birthday. George Braziller, New York. 213 pp.

Luckham, D. (2012). Event processing for business: Organizing the real-time enterprise. Hoboken, New Jersey: John Wiley & Sons, Inc.,. ISBN 978-0-470-53485-4.

Malcolm, C. (1995). The SOMASS system: a hybrid-symbolic and behavior-based system to plan and execute assemblies by robot. In: In Hybrid Problems, Hybrid Solutions. John Hallam, ed. IOS Press. Accessed February 28, 2014: http://books.google.com/books?hl=en&lr=&id=81zIE7Af5LkC&oi=fnd&pg=PA157&dq=Smithers,+T.,+and+Malcolm,+C.A.,+Programming+Robotic+Assembly+in+terms+of+Task+Achieving+Behavioural+Modules,+Journal+of+Structural+Learning&ots=EKvsBULdqZ&sig=Vo5_JJiiv_MBLBOVkYkoaOcqTeg#v=onepage&q=Smithers%2C%20T.%2C%20and%20Malcolm%2C%20C.A.%2C%20Programming%20Robotic%20Assembly%20in%20terms%20of%20Task%20Achieving%20Behavioural%20Modules%2C%20Journal%20of%20Structural%20Learning&f=false.

Meadows, D.H., Meadows, D.L., Randers, J., & Behrens, W. (1972). Limits to growth. Universe Books, NY, 205 pp.

Scolozzi, R. & Poli, R. (2015). System dynamics education: becoming part of anticipatory systems. On the Horizon, 23, 107-118.

Smithers, T. (2002). Synthesis in designing. In. Artificial intelligence in design. J. Gero, ed. Kluwer Academic Press. Netherlands. 3 – 24.

Smithers, T., & Troxell, W. (1990). Design is intelligent behaviour, but what's the formalism?.

Artificial Intelligence for Engineering, Design, Analysis and Manufacturing, 4, 89-98. DOI: http://dx.doi.org/10.1017/S0890060400002286

Veloso, M., Stone, P. & Bowling, M. (1999). Anticipation as a key for collaboration in a team of agents: A case study in robotic soccer. In *Proceedings of SPIE Sensor Fusion and Decentralized Control in Robotic Systems II*, volume 3839, Boston, Sept. 1999. Accessed on July 18, 2015: http://www.cs.cmu.edu/~mmv/papers/spar-spie.pdf

von Bertalanffy, L. (1975). Perspectives on General System Theory: Scientific-philosophical studies. Ed. E. Taschdjian. George Braziller, New York. 183 pp.

Numerical Analysis

Balakirsky, S., Carpin, S., Dimitoglou, G., & Balague, B. (2009). From Simulation to Real Robots with Predictable Results: Methods and Examples. In: Performance Evaluation and Benchmarking of Intelligent Systems. Madhavan, R., Tunstel, E., & Messina, E., eds. Springer, 336pp. Accessed October 20, 2014: http://link.springer.com.ezproxy2.library.colostate.edu:2048/book/10.1007%2F978-1-4419-0492-8

Berghage, R. & Heins, R. (1991). Quantification of Temperature Effects on Stem Elongation in Poinsettia. Journal of the American Society for Horticultural Science, 116: 14-18. Hopper, D., and Hammer, P. (1991). Regression models describing Rosa hybrida response to day/night temperature and photosynthetic photon flux. Journal of the American Society of Horticultural Science, 116:609-617.

Hopper, D., Hammer, P., & Wilson, J. (1994). A simulation model of Rosa hybrida growth response to constant irradiance and day and night temperatures. Journal of the American Society of Horticultural Science, 119:903-914.

Hopper, D., Hammer, P. & Wilson, J. (1996). Comparison of two simulation-based methods for modeling plant growth over time. HortScience, 31:25-28.

Karlsson, M., Heins, R., Erwin, J., Berghage, R., Carlson, W., & Biernbaum, J. (1989). Temperature and photosynthetic photon flux influence chrysanthemum shoot development and flower initiation under short-day conditions. Journal of the American Society for Horticultural Science, 114:158-163.

Paine, C., Marthews, T., Vogt, D., Purves, D., Rees, M., Hector, A., & Turnbull, L. A. (2012). How to fit nonlinear plant growth models and calculate growth rates: An update for ecologists. Methods in Ecology and Evolution, 3: 245-256. doi:10.1111/j.2041-210X.2011.00155.x.

Richards, F. (1959). A Flexible Growth Function for Empirical Use. Journal of Experimental Botany 10 (2): 290–300. doi:10.1093/jxb/10.2.290.

Zhu, C., Unachak, P., Llera, J., Knoester, D., Runkle, E., Xu, L., & Goodman, E. (2014). Robust multi-objective evolutionary optimization to allow greenhouse production/energy use tradeoffs. Acta Horticulturae, 1037:525-532.

Robotics & Related Theory

Baron-Cohen, S. (1995), Mindblindness, MIT Press.

Baron-Cohen, S., Leslie, A. & Frith, U. (1985), 'Does the autistic child have a "theory of mind"?', Cognition 21, 37–46.

Donnett, J. & Smithers, T. (1990). Neuronal group selection theory: A grounding in robotics. In *Advances in Neural Information Processing Systems*, pp. 308-315. Accessed June 5, 2015:

http://papers.nips.cc/paper/204-neuronal-group-selection-theory-a-grounding-in-robotics.pdf
Gibson, J. (1977). The Theory of Affordances. In: Perceiving, Acting, and Knowing. Eds. Shaw, R. and Bransford, J. Hillsdale, NJ. L. Erlbaum. Also: http://en.wikipedia.org/wiki/Affordance.
Haught, J. (2006). Is Nature enough? meaning and truth in the age of science. Cambridge University Press, UK.
Leslie, A. M. (1994), ToMM, ToBY, and Agency: Core architecture and domain specificity, in L. A. Hirschfeld & S. A. Gelman, eds, 'Mapping the Mind: Domain specificity in cognition and culture', Cambridge University Press, pp. 119–148.
Maris, M., & te Boekhorst, R. (1996). Exploiting physical constraints: heap formation through behavioral error in a group of robots. Proceedings of IROS'96: IEEE/RSJ International Conference on Intelligent Robots and Systems, ed. M. Asada (Piscataway, NJ: IEEE Press), 1655–1660. [Swiss robots]

Biological Inspiration and Physical Relations

Compton, A. (1923). A Quantum Theory of the Scattering of X-Rays by Light Elements. Physical Review 21 (5): 483–502. Bibcode:1923PhRv...21..483C. doi:10.1103/PhysRev.21.483.
Feng, L., de Reffye, P., Dreyfus, P., and Auclair, D. (2012). Connecting and architectural plant model to a forest stand dynamics model - application to Austrian black pine stand visualization. Annals of Forest Science, 69:245-255. DOI 10.1007/s13595-011-0144-5.
Fraenkel, G. (1980). On geotaxis and phototaxis in Littorina. in The Organization of Action: A new synthesis. C.R.Gallistel (ed.), Lawrence Erlbaum.
Schrodinger, E. (1967) (originals: 1944 & 1958). What is life? & Mind and Matter. Cambridge University Press, Great Britain. 178 pp.
Yan, H., Kang, M., Reffye, P., & Dingkuhn, M. (2004). A dynamic, architectural plant model simulating resource-dependent growth. Annals of Botany, 93:591-602.

GLOSSARY

agent: specific physical robot or programed process that performs a behavior to attain desired task achievement.

anticipation: creation, formation, formulation, or determination of a suitable process to make a choice from a small set of feasible future scenarios before the outcome is certain for successful behavior beyond mere reactions to items in the niche that leads to desired task achievement with expected immediate or later reward based on perceived fitness matched to the niche.

behavior: observed response of an organism or robotic agent to current niche environmental conditions to create task achievement.

previously from Behavior-based robotics: interaction between the task, environment, and agent with specific capabilities that creates a successful response

behavior-based robotics: an approach that matches robot agent behavior directly to a specific environmental niche condition to promote desired task achievement.

bounded rationality: not all information is known that affects the outcome of a decision.

closed system: system with no inputs from or outputs to the surrounding environment.

fitness: specific suitable agent condition that matches to a specific niche condition.

metaphor: concept of using abstract principles to capture inferences in a formal system without deriving them directly from a congruent natural system by encoding with observation or experimentation. The metaphor principles must be verified to be congruent with the natural system by decoding the abstractions back to the specific instance of the natural system, and observing expected desired behavior and results.

niche: specific environmental surrounding perceivable by the agent. It is a smaller subset of the overall environment or arena that the robot agent is able to exist within, and is a local condition that is relevant to manifestation of a behavior choice.

open system: system with inputs from and/or outputs to the surrounding environment.

percept: creation of internal or mental representations (images or archetypes) using the proximal stimuli derived from distal stimuli.

process: series of events leading to an outcome.

satisficing: a solution is discovered using bounded rationality that is acceptable but not the absolute optimum, since too much time and too many resources would be needed to search across an almost infinite number of possible solutions.

scenario: situation that matches behavior to a niche.

system: collection of interacting components intended to perform a task.

see: closed system, open system.

reaction: direct behavior following a specific niche situation.

reward: confirmation that an action leads to task achievement.

task: desired outcome from a behavior or sequence.

INDEX