A Fuzzy-Logic Autonomous Agent Applied as a Supervisory Controller in a Simulated Environment

Georgios Chrysanthakopoulos, Warren L. J. Fox, *Senior Member, IEEE*, Robert T. Miyamoto, Robert J. Marks, II, *Fellow, IEEE*, Mohamed A. El-Sharkawi, *Fellow, IEEE*, and Michael Healy, *Member, IEEE*

***Abstract*—An unsupervised learning system, implemented as an autonomous agent is presented. A simulation of a challenging path-planning problem is used to illustrate the agent design and demonstrate its problem solving ability. The agent, dubbed the ORG, employs fuzzy logic and clustering techniques to efficiently represent and retrieve knowledge and uses innovative sensor modeling and attention focus to process a large number of stimuli. Simple initial fuzzy rules (instincts) are used to influence behavior and communicate intent to the agent. Self-reflection is utilized so the agent can learn from its environmental constraints and modify its own state. Speculation is utilized in the simulated environment, to produce new rules and fine-tune performance and internal parameters. The ORG is released in a simulated shallow water environment where its mission is to dynamically and continuously plan a path to effectively cover a specified region in minimal time while simultaneously learning from its environment. Several paths of the agent design are shown, and desirable emergent behavior properties of the agent design are discussed.**

***Index Terms*—Autonomous control, clustering, emergent behavior, fuzzy agents, self-reflection, unsupervised learning.**

I. Introduction

An unsupervised problem solver is presented, implemented as an autonomous agent. The agent is a software algorithm with an advanced sensory interface, learning facility, speculation abilities and self-reflection capabilities. It is applied to a path-planning problem in a simulated environment. The agent algorithm, dubbed the ORG (for organism), was designed to address specific existing agent architecture limitations. An overview of autonomous agents is given followed by the design objectives used for the ORG architecture.

A. *Review of Prior Research*

Agents are complete systems that integrate sensors, perception, knowledge, and learning modules to achieve broad goals. All these components operate together and have a simple communication mechanism. An agent is situated in time and space. It deals with problems in an incremental, real-time iterative fashion. Environmental constraints are exploited in agent design and environment features can aid the agent in achieving its goals. For example, if an agent is restricted to move on a two-dimensional surface, it can customize its sensor interface to use the horizon as a reference point.

Side effects arising from the agent's interaction with the environment and from the interaction of its internal modules are often desired and help the system achieve complex goals without complicated heuristics and explicit hard rules. Previous work [1], [2], [4], [9], [10], [12] provides an extensive set of autonomous agent architectures. A good overview of agent architectures, limitations and the fundamental differences among traditional artificial intellignece (AI) approaches is found in [3], [6], and [13].

Any autonomous learning system has to deal with a number of issues, some of which are mentioned here.

1) *Action Selection*: What should the agent perform, from a potentially large set of possible actions to optimize the achievement of its goals? With unsupervised agents this is especially difficult since the goals themselves can be time varying or ill-defined in terms of a single iteration step
2) *Learning from Experience*: The agent should become better at achieving goals with experience.
3) *Scaling to larger problems*: An agent should be able to handle increasingly complex problems that demand a greater amount of knowledge, and to process an increasing amount of sensory information.
4) *Attention focus (reactivity)*: In a rich environment, a large amount of sensory information needs to be processed. The agent should be able to focus its attention on a small number of relevant stimuli so it reacts in real time and makes an optimal action selection based on the few stimuli that were chosen.
5) *Modularity*: Software engineering dictates functional decomposition: components should be designed according to their function, leading to a system easier to build and maintain.

Problems one, two and three are tackled through the simplicity and inherent explanation facility of the fuzzy system used by the ORG (Sections III, V, and VI). The sensor architecture of combining adjustable virtual and real sensor ranges provides a tunable solution for tackling large amounts of stimuli and addresses issues one, four and five (Section IV). Finally, the ORG software design coupled with a new symbol system provides a portable, modular architecture addressing issue five (Section III).

Manuscript received January 28, 2001; revised December 5, 2001 and June 27, 2003.

G. Chrysanthakopoulos is with the Microsoft Corporation, Redmond, WA 98052 USA.

W. L. J. Fox and R. T. Miyamoto are with the Applied Physics Laboratory and the Department of Electrical Engineering, University of Washington, Seattle, WA 98105-6698 USA.

R. J. Marks, II is with the Department of Engineering, Baylor University, Waco, TX 76703 USA.

M. A. El-Sharkawi and M. Healy are with the Department of Electrical Engineering, University of Washington, Seattle, WA 98195 USA.

Digital Object Identifier 10.1109/TFUZZ.2003.822683

B. Multiagent Systems

Agents can exist in a cooperative society of similar agents. Since they are independent they can be assigned different tasks and concurrently tackle different aspects of a problem. Sharing information is an option.

In [10], the authors present a software system for defining multi-agent missions. The work in [10] presents a software environment using object-oriented techniques to define agent behavior and situate the agents in an environment. Rules and module interconnection are defined in computer code *a priori* and then compiled to produce the robotic agent. The ORG instead chooses a flexible representation system based on a combination of fuzzy logic and software data objects. Fuzzy rules and a fuzzy logic inference already provide well established mathematical means for combining multiple human readable rules and dealing with partial or inaccurate sensory information (the fuzzy inference and defuzzification procedures employed by the ORG are described in Appendix A, Section A). Software engineering is utilized to make the agent modular, portable and easier to maintain.

Another approach, used to compose heterogeneous agent implementations and coordinate planning between them is found [16], [17]. The MPA work builds significant infrastructure to support multiple cooperating agents that distribute the planning process. An evolution of a hierarchical task network (the survey in [17] mentions in more detail the planning systems used in distributed planning) is employed as the multiagent planning facility. In [17], the case is made for agents that continually alter their plan due to the dynamic nature of their environment. The ORG architecture presents a single agent interacting with its environment, and with the inclusion of "instincts" and long term goals, with its human operators. A dynamic environment is taken into count with the creation of reactive short terms rules and by incorporating environment feedback in new rules. Unlike the MPA work, however, communication and distributed planning is not addressed in this work. Given the limited scope of this project, the focus has been on the features listed in the following section.

C. ORG Design Features

The main ORG features are summarized as follows.

1) *Simple interface for adding heuristics, explanation facility*: Stimuli, rules and acquired experience is represented as a collection of software objects. Fuzzy logic methods are used to manipulate them, providing an easy methodology for adding domain knowledge and interpreting what the agent has learned at any time during its operation.
2) *Attention focus mechanism that can deal with large number of stimuli*: A fuzzy-logic based stimulus ranking technique (Section VI-B), combined with a range dependent sensory model, enables the ORG to focus its attention to a small number of stimuli (from a large pool of candidates). Range dependent modeling uses two distance thresholds to determine if stimuli should be evaluated (Section IV-A).
3) *Reactive and long-term behavior use*: The agent combines the reaction to multiple stimuli in its immediate environment with the reaction to long term goals defined either before the ORG comes online or during its operation.
4) *Self-Reflection ability*: The ORGs internal state is represented by the same symbols used to describe external stimuli. This allows the ORG to generate a reaction to its current or future state and affect its next state transition.
5) *Platform and application independent interfaces*: The ORG is implemented in a system of software modules that abstract their functions and hide problem or platform specific details. This makes the agent reusable and portable.
6) *Large problem scalability*: The application chosen demonstrates the scaling ability of the ORG since it requires hundreds of stimuli to be evaluated before a reactive decision is made. It also presents unique difficulties since stimulus properties change depending on their relative position to the ORG. The ORG performs well in this scenario and even in its current form meets real time constraints.

II. Application and Simulation Environment Description

To better illustrate the ORG architecture, the simulation of a challenging real world problem is described and used as a running example throughout this work. A description of the problem is given as follows.

> A vessel carrying a sonar/hydrophone array needs to effectively search a large volume of ocean water in order to detect underwater targets. Acoustic models are used to generate sonar performance predictions, which then are used to calculate probabilities of detection for a finite number of points in the ocean.

The ORG is generating the vessel path that meets the physical constraints of the vessel, the time constraints of the task and minimizes the probability a target went undetected. Note that if the environment is known, the ORG simulation can be run before a real vessel goes to sea or it can be run concurrently, taking into account sampled environmental values and other changing parameters.

A. Simulation Environment

A three-dimensional virtual environment represents a 16 000X16000X200 m water volume. A grid point is placed every 1 km in the XY plane and every 15 meters in the Z plane (depth). Fig. 1 is a graphical representation of the simulation environment.

Fig. 1 illustrates the virtual targets in the simulated environment with the ORG path overlaid at the surface. The virtual targets are colored according to the sonar performance at the targets location. The ORG trajectory is colored to illustrate incrementing iteration numbers.

Each grid point is modeled as a static external stimulus and is sensed if it falls within the agent's sensory range. Each stimulus

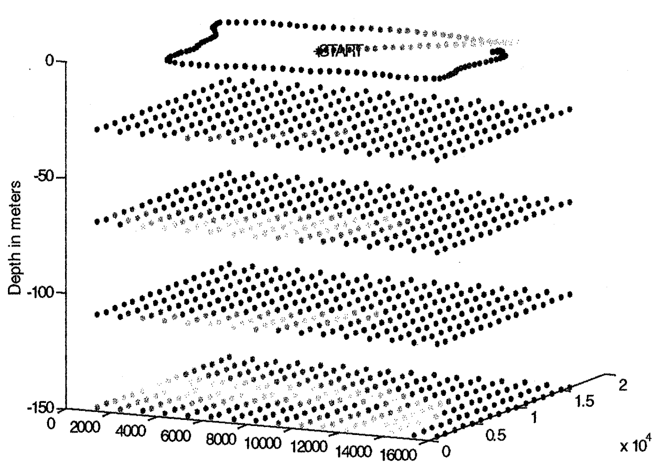

Fig. 1. ORG virtual environment.

has a number of dynamic properties that change depending on its position relative to the agent and the number and duration of prior encounters. Table I lists these properties.

The signal excess property above reflects the sonar performance prediction. This value (and the associated probability of no detection) changes for each stimulus depending on the ORGs distance and orientation. This adds a challenge to the path-planning problem since the agent's perception of the environment changes as it moves. For example the same grid locations (and the stimuli representing them) will look entirely different to the ORG if it approaches them from two different angles.

B. Application Parameters

The agent assumes the role of the surface vessel searching the simulated environment, carrying a sonar/hydrophone array through a large water volume. The *task* is to form a path that minimizes the probability of a target being present and undetected (P_{ND}) for all grid points. The constraints can be summarized as follows:

1) arbitrary velocity changes are not allowed and are constrained by physics;
2) the simulation objective must be achieved within a certain time.

III. Overview of Internal Architecture

A. Algorithm Flow

The ORG is implemented as an object oriented program with a primary iteration loop. A single iteration step is illustrated in Fig. 2. The action selection module at the bottom of Fig. 2 performs the last step in the iteration. The next iteration starts after the ORG collects environment information and passes the environment data to the sensor module at the top.

The bidirectional arrows indicate that even if the ORG architecture is generally a feed forward loop, there exist mini-loops that feed back information. The core module is invoked multiple times from within other modules and there is no strict hierarchy among components (i.e., this is a flat network).

B. Internal Representation and the ORG Symbol System

Before the sensory module and the learning mechanism is presented, the manner by which the ORG stores information internally is explained. This section introduces the software symbols utilized to create internal representations of stimuli and the ORG itself.

The *physical symbol system hypothesis* [11] proposes that a physical symbol system has the necessary and sufficient means for general intelligence. Newell differentiates between the *symbols* (the abstraction of physical phenomena) and *tokens* (the physical instantiations). A *symbol system* has the following [3]:

- *Memory*: to store the symbol information;
- *Symbols*: to provide a pattern to match or index remote token information;
- *Operators*: to manipulate symbols;
- *Interpretation*: to derive operations from the symbols.
- *Capacities* for:
  1) sufficient memory,
  2) composability (the operators may make any symbol structure),
  3) interpretability (the symbol structures be able to encode any meaningful arrangement of operations).

A versatile symbol system and symbolic architecture that minimizes memory utilization and allows for efficient lookup of knowledge is proposed here. The ORG defines data structures to represent stimuli, fuzzy rules and its own internal state. The data objects and the methods used to manipulate them create the ORG symbol system. The code implementation is the fixed symbolic architecture

The ORG software architecture defines a set of data structure objects used as primitives for information exchange and knowledge acquisition. The primitives are used as flexible containers capable of representing a wide range of objects with varying numbers of properties. Fig. 3 shows the object hierarchy. Here is a sample C language declaration of the object types (the ORG code is implemented in MATLAB 5 m-file language but C equivalent code is used here for clarity).

```
typedef struct _PROPERTY_OBJECT {
    DWORD ObjectType;
    BYTE    Index [MAX_PROPERTIES];
    BYTE    PropertyValues [MAX_PROPER-
TIES][MAX_SIZE];
} PROPERTY_OBJECT;

typedef struct _MEMORY_OBJECT {
    PROPERTY_OBJECT ConditionObject;
    PROPERTY_OBJECT EffectObject;
    BYTE    MergeCount;
    BYTE    LastIterationUpdated;
} MEMORY_OBJECT;
```

The `PROPERTY_OBJECT.PropertyValues` matrix contains the values for each property vector (row N, for property N). Each property can optionally have an associated time derivative, which is treated as a separate property and assigned an index

TABLE I
LIST OF PROPERTIES IN EACH STIMULUS

Property	Description	Dependencies
Signal Excess (SE)	Generated by a sonar acoustic model. High values translate to high probabilities of detection.	Re-calculated each time the org moves since SE detectability depends on the ORGs orientation and distance
Probability of target Not Detected (P_{ND})	Conditional probability calculated using Bayes theorem and a target strength distribution function.	Depends on *SE* value
Distance	Distance of a stimulus to the ORG	Depends on the ORG *position* property
Position	X,Y,Z coordinates	None
Theta	Angle between vector to the stimulus and current velocity vector	Depends on ORG direction

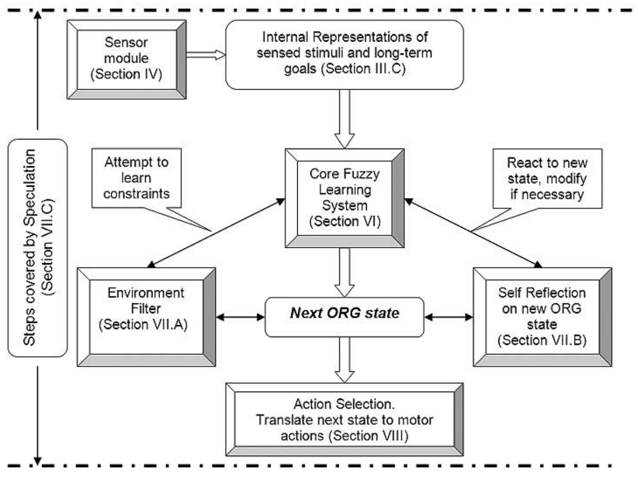
Sensor module (Section IV)
Internal Representations of sensed stimuli and long-term goals (Section III.C)
Attempt to learn constraints
Core Fuzzy Learning System (Section VI)
React to new state, modify if necessary
Steps covered by Speculation (Section VII.C)
Environment Filter (Section VII.A)
Next ORG state
Self Reflection on new ORG state (Section VII.B)
Action Selection. Translate next state to motor actions (Section VIII)

Fig. 2. Single iteration step of the ORG loop.

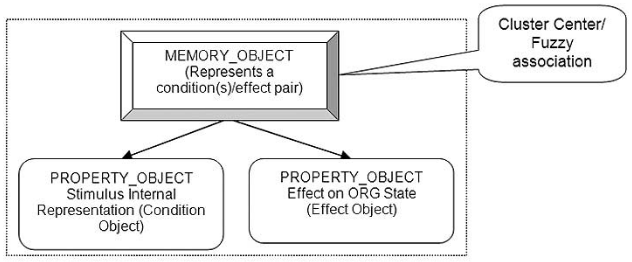
MEMORY_OBJECT (Represents a condition(s)/effect pair)
Cluster Center/ Fuzzy association
PROPERTY_OBJECT Stimulus Internal Representation (Condition Object)
PROPERTY_OBJECT Effect on ORG State (Effect Object)

Fig. 3. Symbol hierarchies.

number of N + 1 in PROPERTY_OBJECT.Index. (N is the column index assigned to the original property, N + 1 is the column index of its first derivative). Derivative properties used for the application described are dP_{ND}/dt, $dDistance/dt$ (speed) *and* $dPosition/dt$(velocity). A property is considered *active* if PROPERTY_OBJECT.Index[N] is set to one, where N is the assigned property number. The properties active in the internal representation correspond to the physical properties present in the stimulus. The same symbol (PROPERTY_OBJECT in this case) can thus be used to describe different types of stimuli, with different physical properties.

C. Representation of Stimuli and the ORG State

A stimulus is represented using a PROPERTY_OBJECT. The internal ORG state is also represented using the same object but its ObjectType field is different (see Table II) and has a different set of active properties. This allows the core module (Section VI) to filter learned associations intended for self-reflection (Section Section VII-B) and not use them to generate reactions to stimuli.

Sample "C" code for defining the internal ORG state is given as follows:

```
ORG.ObjectType
        = STIMULUS_INTERNAL|STIMULUS_REACTIVE.
```

An external stimulus (the virtual targets in each grid point) is defined as

```
Stimulus.ObjectType
        = STIMULUS_EXTERNAL|STIMULUS_REACTIVE.
```

IV. SENSOR MODULE

Information about the environment gathered from various sensors flows first through the ORG sensory module. Separate functions within the module process raw information and create standard data structures (Section III-B) to represent discrete stimuli. A stimulus can appear discrete under a low-resolution sensor and appear as multiple interconnected objects under a more advanced sensor. In the simulation discussed here, all external objects are discrete with a fixed number of observable properties (Table I).

Whenever the ORG encounters a stimulus it creates an **internal representation** of that object using the PROPERTY_OBJECT structure. Due to learned experience or prior knowledge in the form of initial rules, the ORG might have *a reaction* to that representation. The reaction (described in Section VI-B) is the effect the stimulus will have on the next ORG state. That effect is also modeled using a PROPERTY_OBJECT and associated with the internal representation of the stimulus using a MEMORY_OBJECT. The combined reaction to all stimuli forms the *state transition* from the current ORG state to the next.

A. Stimulus Filtering

The *distance* property is compared to a threshold to determine if the object is sensed in the current iteration and if its internal

TABLE II
INTERNAL REPRESENTATIONS OF OBJECTS

PROPERTY_OBJECT.ObjectType (Bit field)	Description
STIMULUS_EXTERNAL	Object is external to the ORG
STIMULUS_INTERNAL	Object represents an internal ORG state
STIMULUS_REACTIVE	Object is real (actively sensed)
STIMULUS_LONGTERM	Object compatible with long-term rules (learned condition/effect pairs). Its a representation of a stimulus previously encountered or of a desirable ORG state

TABLE III
EXPLANATION OF RANGE DEPENDENT SENSOR BEHAVIOR

Stimulus/Internal Representation distance (*d*) to ORG	Sensor behavior
$d < T_{\text{physical\_range}}$	Stimulus properties determined, internal representation properties can be determined and modified
$T_{\text{physical\_range}} < d < T_{\text{virtual\_range}}$	Stimulus properties determined, internal representation available but can not be modified
$d > T_{\text{virtual\_range}}$	Stimulus can not be sensed, internal representation is not made available

representation will be made available to the ORG. For this simulation the *distance* property has an infinite range, which means that no matter how far away an object is its distance can be determined. As the ORG encounters stimuli it creates data objects to represent them internally. In the simulation presented here, each stimulus is a virtual target placed at a fixed location on a grid. When the object falls within range, its internal representation becomes part of an internal map of the environment. Not all properties active in the representation of the stimuli, (P_{ND} is one example), have a physical equivalent. They are, instead, virtual properties that are only present in the data object used for the internal representation of the stimulus. This is an application dependent behavior and, in the current simulation, is needed to keep track of the cumulative P_{ND} associated with all virtual targets the ORG encounters.

The ORG employs two sensor thresholds to distinguish between the actual sensor limitations ($T_{\text{physical\_range}}$) and when to make the internal representation of a stimulus "available" ($T_{\text{virtual\_range}}$). Internal representation properties are updated only for stimuli that are within a close range $T_{\text{physical\_range}}$. In a real environment (vs. the simulation discussed here) $T_{\text{physical\_range}}$ is reduced to the physical limit of the particular sensor, or even smaller (here, it is set to 6 km). This sensor modeling is summarized in Table III and utilizes *spatial locality to select a small number of relevant stimuli.*

The ORG is capable of creating internal representation of stimuli from the following two sources:

1) the environment—with sensors providing the data describing each object;
2) a model—with a preinitialized environment filled with virtual objects, as in a simulation.

In the simulation discussed here, if the human designer has model-supplied data or prior knowledge of the area being searched, he/she can preinitialize a virtual environment with objects so that the ORG can sense and create internal representations. Then the ORG merges both the virtual environment with the real world representations to make decisions. This approach works for the specific application and might not be possible with other applications due to the *frame problem* discussed in [3] (section on architecture issues). The frame problem arises when the agent cannot easily associate an internal representation of an external object either because the object is moving or because the sensor information is not sufficient to distinguish it from others. In the simulation the static virtual targets are placed on a regular grid pattern, allowing the ORG to associate internal representations of stimuli created at some point in the past, with currently observable virtual targets. In [9], multiple agents pursue each other but the frame problem is also addressed in the simulation by assigning unique color to each agent.

V. INSTINCTS

Instincts are abstract rules that influence ORG behavior throughout the simulation. In this application instincts instruct the ORG how to react to external stimuli and long-term goals. The human designer forms instincts by specifying a MEMORY_OBJECT and then invoking the Core module in order for the ORG to learn the new association. The learning function is described in the following section. Once an instinct becomes part of the ORG, it can be modified during the simulation, just like other knowledge. The MEMORY_OBJECT representing the instinct translates to a fuzzy linguistic rule. The antecedents refer to properties in the internal representation of a stimulus, the consequents describe the effect that sensing this stimulus will have on the ORG state. One of the instincts used in this simulation is described as follows.

IF P_{ND} is VERY LARGE **and** Induced change in ORG direction (dTHETA/dt) due to this stimulus is VERY SMALL **and** distance to stimulus is SMALL *THEN* attempt LARGE DECREASE in distance to stimulus with the *intent* of LARGE DECREASE in P_{ND} in the location of the stimulus.

Instincts can provide a performance indicator and the ***notion of intent*** if they specify the derivative of an internal ORG property and the desirable change direction (negative or positive). The intent property must be part of the Effect portion of the instinct and must not map to a motor function. This is how the ORG discovers intent properties. In this context, "intent" is a vague effect property that helps the ORG determine the motivation behind the instinct.

The use of instincts highlights one desirable feature of lazy learning methods (see [7, p. 245]): Vague rules representing prior knowledge of the problem domain are utilized to sustain performance when available training instances are scarce or not available. An example of an instinct is given as follows in MATLAB5 script language.

```
%
% Define condition property object
% Active properties we look for in the
stimulus are
% 1) its current probability of detec-
tion
% 2) the deviation it would cause to
our current velocity
% 3) its distance from the ORG
% The variables are defined as fol-
lows:
% PROB = probability of target present
(PND)
% THETA = angle between the ORG ve-
locity vector and the
%     direction vector to the stimulus
(THETA+1 is the first
%     time derivative)
% DIST = Euclidean distance to the
stimulus
% FUZZY_SET_xxxx = Fuzzy set support
(center) number. It
%          characterizes the fuzzy set
to use. This can
%          be used if a crisp number
is not desired.
% pI is a variable of type PROP-
ERTY_OBJECT used to specify the
% condition portion of the instinct
% pO is a variable of the same type,
used to define the effect
% portion of the instinct. The in-
stinct is then defined by
% variable m, of type MEMORY_OBJECT.
pI = PROPERTY_OBJECT;

pI.Index(1,[PROB THETA+1 DIST ]) = [1
1 1];
pI.Type = bitor (STIMULUS_EX-
TERNAL,STIM_RT);

pI.Values(PROB) = FUZZY_SET_LARGE_POS-
ITIVE;
pI.Values(THETA+1) = FUZZY_SET_ZERO;
pI.Values(DIST) = ORG.Con-
straints.NearSensorRange;
%
% Define the effect property object.
% Only one motor action is defined:
Decrease distance
% One intent property is defined: De-
crease probability target is
% present
%
pO = PROPERTY_OBJECT;
pO.Index(1,DIST+1 PROB+1]) = [ 1 1];
pO.Values(DIST+1,1) =
FUZZY_SET_LARGE_NEGATIVE*
  SIM.ORG.PropertyObject.RangeHi
(DIST+1);
pO.Values(PROB+1,1) = SET_LARGE_NEGA-
TIVE;

% Create rule association in the form
of a MEMORY_OBJECT

m = MEMORY_OBJECT;

m.P = pI;
m.Effect = pO;

% Call the CORE_LEARN function to
cluster the memory object to
% the ORG's existing knowledge

Core(CORE_LEARN,m);
```

VI. Core Fuzzy System

The core fuzzy system is a variable-input–variable-output fuzzy engine that uses the data structures defined above to determine the relevant information content to be used from each learned fuzzy association. A fuzzy association is a rule comprising of an input and output pair. The rule is in human readable form since it can be translated to a linguistic rule in the form described in Section V. A clustering algorithm is used to compress information and merge similar pairs with each other. ORG modules invoke the core fuzzy module (central block in Fig. 1) in the following modes of operation.

1) CORE_LEARN—The caller module supplies a `MEMORY_OBJECT` with a condition and effect object.

The core learns this new association through fuzzy clustering

2) CORE_REACT—The caller module supplies a MEMORY_OBJECT with only a condition object (the effect is left empty). In response the core generates a reaction to this condition using all relevant existing associations. The reaction is returned in the form of an effect PROPERTY_OBJECT. If there are no relevant associations in the ORG's memory, an empty (null) reaction is generated and the ORG does not modify its behavior due to this condition object.
3) CORE_RECALL—The caller module supplies a MEMORY_OBJECT with only a condition object (the effect is left empty). The core will attempt to find all similar condition objects and return them in a list, using a fuzzy membership indicator (discussed in Section B) to rank them in descending order

Note that the Core module could potentially use a different algorithm for learning and later recall condition/effect pairs. Fuzzy logic is used because it provides an explanation facility. Learned rules are in human readable form and for each input presented to the core fuzzy engine, the human designer can trace which rules, and to what extent they contributed to the output ([14] and [15] are highly recommended for readers not familiar with fuzzy logic).

A. Learning New Associations (CORE_LEARN)

This function is used to add a new MEMORY_OBJECT to the existing rule base. It uses the clustering procedure described in Appendix A to add a new condition/effect pair (MEMORY_OBJECT). New clusters are created or old ones absorb the new MEMORY_OBJECT as needed in order to fit the finite memory implementation of the ORG. Learning is described in more detail in Section VII.

B. Reacting to a Stimulus

The sensor module (top block in Fig. 1) creates a condition object for each stimulus within $T_{\text{virtual\_range}}$, invoking the core (module B) for a fuzzy reaction to this condition. A fuzzy reaction is the output produced by evaluating the input through (2) in Appendix A for each rule in the fuzzy rule base. The core then returns a fuzzy output object [Appendix A, (3)], which is combined to form a condition/effect pair in the form of a MEMORY_OBJECT. Each pair is then ranked in descending order based on the total fuzzy membership value its condition generated when compared to all existing rules. That value is called the ***fuzzy rule base membership indicator*** and is calculated using standard fuzzy logic inference techniques [Appendix A, (2)].

High membership indicators indicate that the particular stimulus matched well to the already learned associations (such as instincts) in the core. The effect objects of the highest ranked N pairs are then averaged together and used as the *combined reaction* to the environment for this iteration. N is a tunable parameter in the ORG state and is usually a small percentage of the number of stimuli within $T_{\text{virtual\_range}}$. Using the fuzzy membership indicator of how relevant existing rules are to the current stimulus, combined with range dependent filtering, described in Section IV-A, comprises *the ORG attention focusing mechanism.*

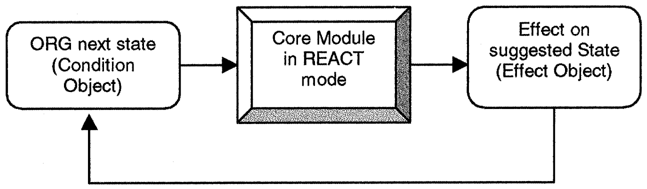
ORG next state (Condition Object)
Core Module in REACT mode
Effect on suggested State (Effect Object)

Fig. 4. Process of self-reflection.

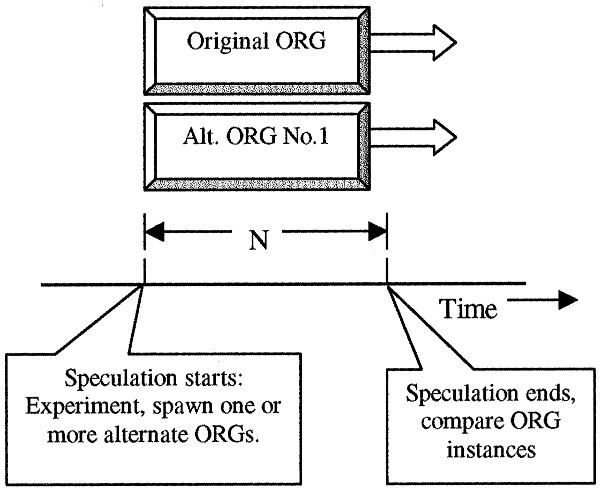
Original ORG
Alt. ORG No.1
N
Time
Speculation starts: Experiment, spawn one or more alternate ORGs.
Speculation ends, compare ORG instances

Fig. 5. Speculation process.

C. Long-Term Goals

Long-term goals motivate the ORG to escape local minima and provide the means for human objectives to be incorporated into decision-making. A long-term goal is defined by the following.

1) A PROPERTY_OBJECT describing the long-term stimulus. This can describe an object, physical or virtual, with any number of active properties. It can also be a desired future internal ORG state
2) A fuzzy association between a condition and an effect (a MEMORY_OBJECT) that tells the ORG how to react to the object in 1).

During ORG initialization (before the iteration loop starts) a list of all known long-term objectives is created and used during the iteration loop to generate a cumulative long-term reaction. The aggregation of ORG reactions to each long-term stimulus is done in the same fashion as with the sensory stimuli, but with no ranking. This means all long-term objectives are considered.

1) Dynamic Long Term Goal Generation Utilizing Sensed Stimuli: The highest ranked internal representation of a stimulus with a distance d to the ORG, which satisfies the inequality

$$T_{\text{physical\_range}} < d <= T_{\text{virtual\_range}}$$

is added to the long-term list. The process of adding internal representations of desirable stimuli, that satisfy the above inequality, to a volatile list for future evaluation, is the ORGs dynamic long-term generation facility. In this simulation, "desirable" are stimuli with a high P_t property, due to the instincts

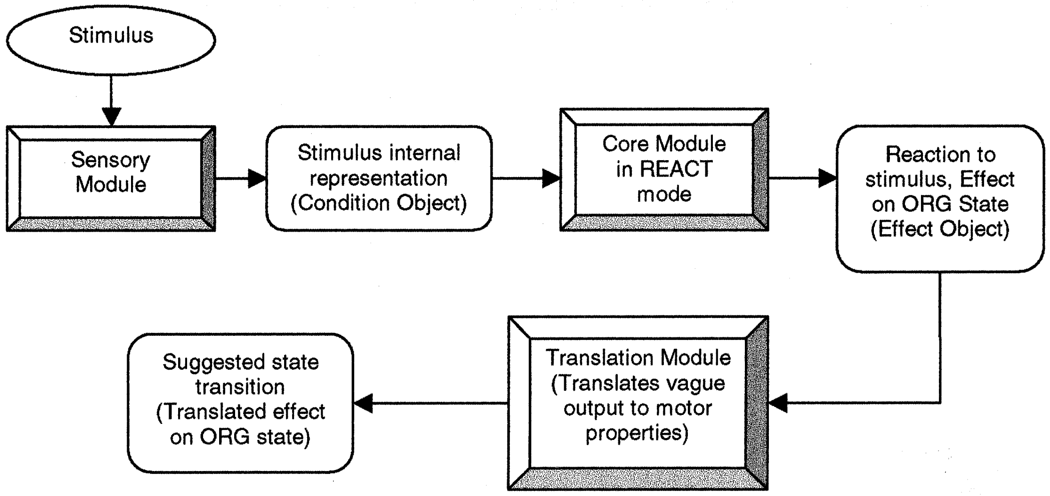
Stimulus
Sensory Module
Stimulus internal representation (Condition Object)
Core Module in REACT mode
Reaction to stimulus, Effect on ORG State (Effect Object)
Translation Module (Translates vague output to motor properties)
Suggested state transition (Translated effect on ORG state)

Fig. 6. Translation of a stimulus reaction, to an ORG state transition.

used. The list of stimuli is used to keep the ORG reacting to some input in situations when no desirable stimulus is within sensor range. This reduces the chance the ORG stagnates (no longer moves or moves in circles). Objects that have come within close range in the past will stop being considered in the long-term list because the ORG can affect their P_{ND} values making them less relevant to one of the current instincts. This eliminates the need of keeping track when a goal generated with this means has been "achieved."

D. Clustering Algorithm

A variant of the sequential fuzzy c-means algorithm [5], with no predetermined number of clusters, is used to classify rules based on their condition (input) PROPERTY_OBJECTs. Each cluster is the association of an input PROPERTY_OBJECT (for example, a stimulus) to an output PROPERTY_OBJECT (the effect the stimulus has on the ORG). The input object is *condition* portion and the output object is the *effect* object. The fuzzy membership of the input condition to the condition object of the cluster is used as the clustering threshold. Clusters are represented by MEMORY_OBJECTs and are created dynamically based on a global expansion threshold T_e [defined in Appendix A, (1)]. A maximum number of clusters can optionally be specified to force any new associations to merge with an existing cluster. The clustering method (described in Appendix A) is a data driven approach with certain thresholds determining the maximum number of clusters after instances are classified.

VII. Learning

The ORG can learn by

1) observing the changes imposed by the environmental filter and learning environmental constraints;
2) speculation and experimentation.

A. Environment Filter

As mentioned earlier (Section IV-A), the combined reaction to external stimuli is used to generate an effect on the ORGs current state. The effect object contains the state transition matrix for the ORGs motor functions. Motor functions are described by properties only active in the ORG internal state. The state transition matrix is comprised of derivative properties that express the transition of each property from its current state to the next (note that state changes only between iterations). Before the state transition matrix is applied to the current ORG state, the simulation code ensures that the transition will not violate physical laws and constraints of the vessel the ORG is emulating. To guarantee this, each state transition is filtered through an *environment filter module.*

The filter module emulates the natural world. It uses physical models and modifies the state transition matrix to fit the constraints of the environment and of the vessel the ORG is controlling. The modified state transition is called the *filtered state transition* and is compared with the original state transition. If they differ, a condition/effect association is created in the form of a MEMORY_OBJECT with the intended state transition associated with the filtered state transition. By repeatedly learning similar cause and effect relationships the ORG can use the learned fuzzy associations to predict environmental constraints and self-reflect (see following section) on its intended state transition.

B. Self Reflection

The ORG makes decisions based on the following input:

1) sensory information from its current environment (Section IV-A);
2) long-term objectives (Section VI-C);
3) its own internal state.

Each state transition generated by the core fuzzy module should adhere to physical constraints. The ORG addresses this by learning prior modifications to its state transition, imposed by the environment filter, and then utilizes that knowledge each time it attempts to modify its state. That process is called self-reflection since the ORG generates a reaction to the internal representation of its proposed next state. That reaction is then applied to modify the proposed state change before it is translated to motor functions (Fig. 4).

The self-reflection feedback loop is run only once per iteration.

C. Speculation and Experimentation Facility

Speculation is part of the learning process and is used to evaluate new rules and determine appropriate motor function mappings for reactions generated by the core module. The ORG starts the speculation procedure by creating one or more alternate versions of itself through experimentation. It then runs the alternate version(s) for N iteration steps, reacting to a snapshot of the environment and performing N "soft" state-transitions, predicting the sensory input at each step. At the end of the speculation run, the ORG compares the performance of the alternative ORG(s) with the original ORG (which also run for N time steps in parallel) and chooses the one that performed better, according to the instincts of the original ORG (Fig. 5). The simulation environment makes speculation feasible since its possible to predict sensory input as the ORG moves in the simulation space.

This strategy differs from those presented in [2] since it compares the alternative system as a whole, not just the new rules created by experimentation. Emergent properties that arise during the N iteration steps can then be observed and utilized in determining the usefulness of the experiment.

D. Experimentation Strategy

The ORG experiments in the beginning of speculation by creating new condition/effect associations or by slightly modifying existing knowledge. Each new fuzzy association (represented by a MEMORY_OBJECT) is then presented to the core fuzzy module, operating in the CORE_REACT mode. If a similar association exists in the core, a high fuzzy membership indicator will be returned and the new fuzzy association will be discarded. If the new fuzzy rule is kept however, the ORG is cloned (see Fig. 5) creating an exact replica of itself. The clone then adds the new rule to its fuzzy rule base (see Appendix A, Section B for the process of clustering a new rule) and the clone is evaluated as a whole for N iteration steps.

VIII. Action Selection

The sensor module invokes the core module to generate a reaction using all the learned associations and instincts within the fuzzy rule base. The reaction is an effect data object. The properties present in the effect might not map exactly to ORG motor functions since they were generated by using a number of vague rules (instincts) and acquired knowledge. Thus, an application specific function is called to perform a translation (translation module in Fig. 6). If the agent is not given, the motor function to action mapping, it can learn how to create that mapping by experimentation. The instincts provide the means to determine which coordinated motor action is required, given a vague effect property.

While experimenting, the ORG can determine which motor function resulted in the desired effect described by the instinct. For example, the instinct in Section V calls for the ORG to decrease distance to a stimulus. If the ORG creates a velocity vector that brings it closer to the stimulus (during experimentation), this motor action will then be associated with the effect property in the instinct.

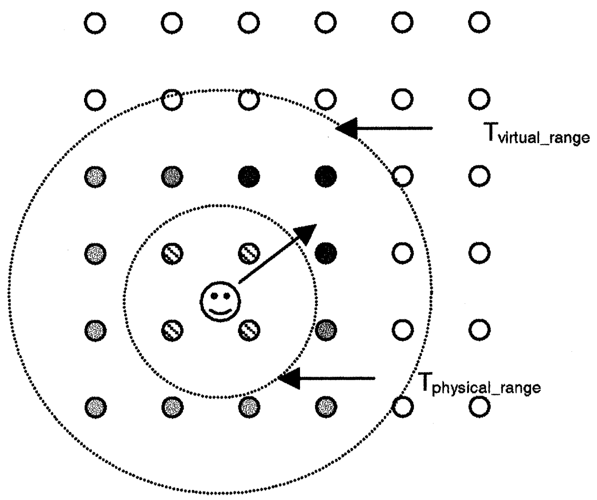

Fig. 7. Tendency to cover new area.

IX. Discussion of Results

The ORG produced several paths through the simulated environment that met the constraints. Different environment sizes, starting positions and initial instincts were tried. Also, different features and capabilities of the ORG were modified to observe the effect they had on the ORG path.

A. Side-Effects Resulting in Emergent Behavior

By utilizing different instincts and by modifying the values for $\mathrm{T}_{\mathrm{physical\_range}}$ and $\mathrm{T}_{\mathrm{virtual\_range}}$, interesting side effects emerged. The side effects described in the following sections emerged because of the interaction of the ORG with the specific simulation environment described in Section II.

Three different instincts were used (in descending order of complexity) in the simulations described in the following sections.

1) *IF* P_{ND} is VERY LARGE, *THEN* attempt LARGE DECREASE in distance to stimulus with the *intent* of LARGE DECREASE in P_{ND} in the location of the stimulus.
2) *IF* P_{ND} is VERY LARGE **and** Induced change in ORG direction (dTHETA/dt) due to this stimulus is VERY SMALL, *THEN* attempt LARGE DECREASE in distance to stimulus with the *intent* of LARGE DECREASE in P_{ND} in the location of the stimulus.
3) *IF* P_{ND} is VERY LARGE **and** Induced change in ORG direction (dTHETA/dt) due to this stimulus is VERY SMALL **and** distance to stimulus is SMALL, *THEN* attempt LARGE DECREASE in distance to stimulus with the *intent* of LARGE DECREASE in P_{ND} in the location of the stimulus.

1) Motivation to Always Keep Moving Toward Unvisited Areas: In the discussion of the sensory interface we mentioned that the virtual properties of only the stimuli within $\mathrm{T}_{\mathrm{physical}_{\mathrm{range}}}$ are changed. In this simulation this translates to

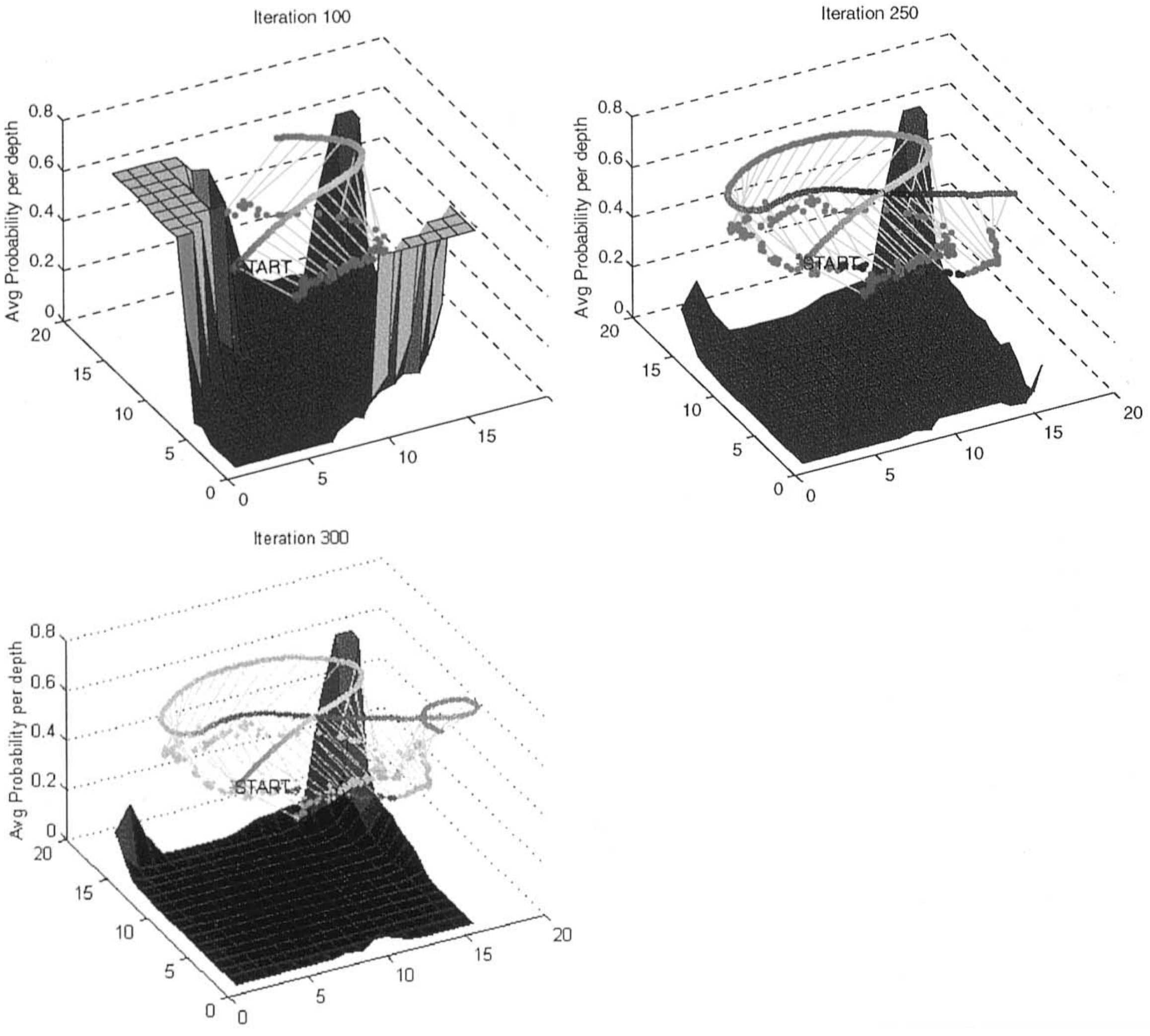

Fig. 8. ORG run with instinct 1 and with $\mathrm{T}_{\text{physical\_range}} = 5000\ \mathrm{m}$, $\mathrm{T}_{\text{virtual\_range}} = 8000\ \mathrm{m}$.

the probability values being reduced since they came within close range of the ORG. Those stimuli thus become "un-attractive" (dashed objects in Fig. 7) and cease to influence the ORG, effectively pushing the agent toward any unaffected objects that lie in the area between $\mathrm{T}_{\text{physical\_range}}$ and $\mathrm{T}_{\text{virtual\_range}}$ (objects with solid gray or black colors). Due to inertia a further bias exists that pushes the ORG toward the objects that lie in its current direction of movement (solid black objects). This behavior resulted with any of the three instincts due to one common condition found in all of them: *decrease distance to stimulus* based on probability.

As Fig. 7 illustrates, the ORG has a bias toward stimuli that lie in its current path where distance satisfies the inequality is $\mathrm{T}_{\text{physical\_range}} < d < \mathrm{T}_{\text{virtual\_range}}$.

2) Rapid Probability Reduction Versus Exhaustive, Longer Path: A side effect of the attention-focusing scheme employed by the agent was that the ORG moved fast through the entire area (thus reducing rapidly the total probability a target was present and un-detected). However a very small portion of the search area was not covered. This is a typical result present in all simulation runs and is illustrated in Fig. 8, where only the upper right corner still has a large average probability for that depth column.

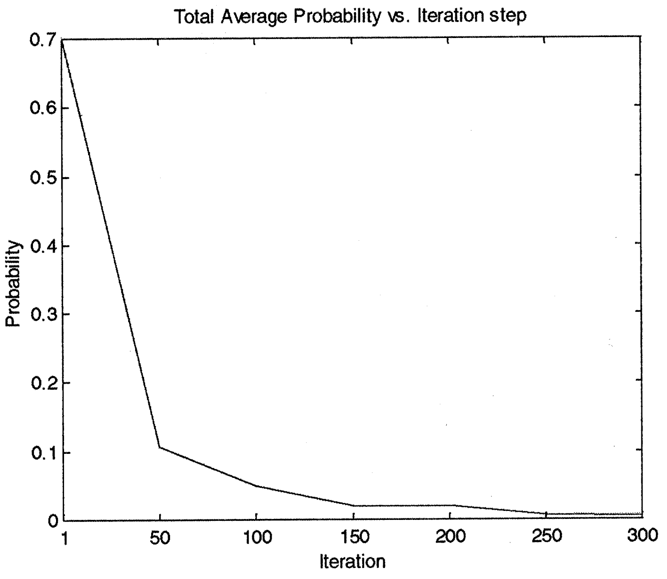

Fig. 9. Rapid probability reduction.

The position of the ORG during each iteration step is represented by a point in the path overlaid on top of each mesh in

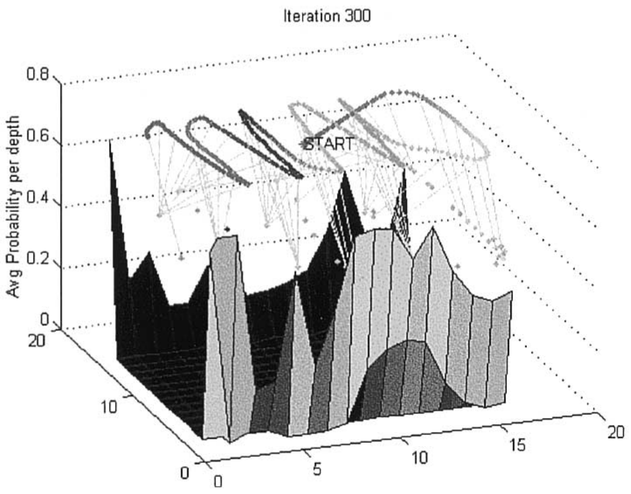

Fig. 10. Results when instinct 2 was used.

Fig. 5. Each iteration corresponds to 30 seconds of real time, with a constant ORG speed of 10 knots. The ORG starts on the bottom left corner of the first snapshot and then follows a diagonal path through the search area turning left when it reaches the upper right corner. The path is color-coded to reflect the time progression. The three dimensional colored mesh under the path represents the average probability of target present (P_T) per depth column. The thin lines connect points in the ORG path to the centroid of the ten most attractive stimuli at each iteration step. The last plot in in Fig. 8 illustrates the diminished returns after iteration 250: the ORG has already reduced the overall probability and is focusing in the lower right corner (red loop in the path) in an effort of diminished returns.

Fig. 9 is plot of the total probability reduction vs. iteration for the path in Fig. 8. In the first 50 iterations, the ORG had reduced the overall probability by more than 80% of its initial value.

3) Dependency on $\mathrm{T}_{\mathrm{virtual\_range}}$ *and the Number of Stimuli Chosen Per Ranking:* The ORG employs two sensor thresholds (Section IV) to distinguish between the actual sensor limitations $(\mathrm{T}_{\mathrm{physical\_range}})$ and when the internal representation of a stimulus should become available $(\mathrm{T}_{\mathrm{virtual\_range}})$. If the $\mathrm{T}_{\mathrm{virtual\_range}}$ is expanded to a number close to the size of the simulated environment, the ORG sees all grid points from any position and hundreds of stimuli get processed during each iteration step. This is desirable if the human designer wants the ORG to take advantage of prior knowledge of the entire environment, while at the same time reacting to stimuli within close range.

Instinct 1 gives no preference to distance or angle from the current ORG path, evaluating stimuli only based on their current probability. This creates erratic paths since stimuli far apart (and ranked high enough to be chosen with the top N stimuli) generate conflicting reactions from the ORG. Even when instinct 2 is used, unsatisfactory results are obtained (Fig. 10). Note that for the path in Fig. 10 each iteration step represents 90 s (previous paths used 30 s) so this path is especially time consuming.

Fig. 10. is from an ORG simulation with $\mathrm{T}_{\mathrm{virtual\_range}} = 14\ \mathrm{km}$ and instinct 2 utilized. Fig. 11 is the resulting ORG path when instinct 3 was utilized and $\mathrm{T}_{\mathrm{virtual\_range}} = 14\ \mathrm{km}$, $\mathrm{T}_{\mathrm{physical\_range}} = 3\ \mathrm{km}$. With all other parameters remaining the same the resulting path was much shorter and smoother.

The attention focusing mechanism depends highly on the fuzzy membership indicator computed for each stimulus in the sensor module. Utilizing an instinct with more input condition properties (instinct 3) makes the fuzzy membership indicator sensitive to more information per stimulus. Stimuli that looked "desirable" with instinct 1, now ranked very differently, helping the ORG focus better. These results strongly suggest that the complexity of the instinct must increase with the amount of stimuli being processed.

B. Effects of Learning on the ORG Path

In the previous section, we presented the ORG behavior that emerged out of the simple instincts and the ORG architecture. Environment constraint learning and speculation were not enabled. In this section we will discuss the resulting ORG paths when learning is utilized during the simulation.

When learning was enabled the attempted ORG state transitions adhered more to the physical constraints, reducing the changes imposed by the environment filter. The information contained in the ORG core memory module reflected which type of fuzzy associations the ORG had learned at the end of the simulation. Since the ORG memory has a finite size, when the memory limit was reached, the clustering algorithm would either replace fuzzy associations entirely or modify existing ones. Fig. 12 shows the number of clusters in the ORG core memory, updated or replaced during the simulation. If a cluster was replaced, this shows when the replacement took place. Note that cluster 1 is occupied by the instinct, and it's the only one that did not get replaced by the fuzzy associations produced from learning.

Fig. 12 shows the iterations during which the core clusters were last updated for the simulation in Fig. 15. Fig. 13 and Fig. 14 contrast the ORG path between simulations with learning external constraints enabled and one with learning disabled. All other simulation parameters were identical. Utilizing learning to alter the next state transition produced a smoother, shorter path as illustrated by Fig. 14. The probability reduction was nearly identical between the two runs. Each ORG path missed stimuli on one of the corners primarily due to the very short physical range (3 km). Fig. 15 is the simulation result when the physical range was set to 6 km and physical constraint learning was enabled. The entire space was covered and the total average probability was reduced to near zero.

Fig. 13 is the resulting ORG path with $\mathrm{T}_{\mathrm{physical\_range}} = 3\ \mathrm{km}$, $\mathrm{T}_{\mathrm{virtual\_range}} = 16\ \mathrm{km}$ and with instinct 3 utilized. Learning from constraints disabled.

The ORG path with same simulation parameters as Fig. 13 but with learning from external constraints enabled produces a smoother and shorter path (Fig. 14.)

Fig. 15 is another illustration of the posistive effects of learning. For this ORG path $\mathrm{T}_{\mathrm{physical\_range}} = 6\ \mathrm{km}$, $\mathrm{T}_{\mathrm{virtual\_range}} = 16\ \mathrm{km}$ and instinct 3 was used.

C. Robustness

The outcome of many simulation runs revealed that the ORG produced desirable paths independent of its starting position and orientation. The simulation run of Fig. 16 has the same parameters as that of Fig. 15 except the ORG starting position. The

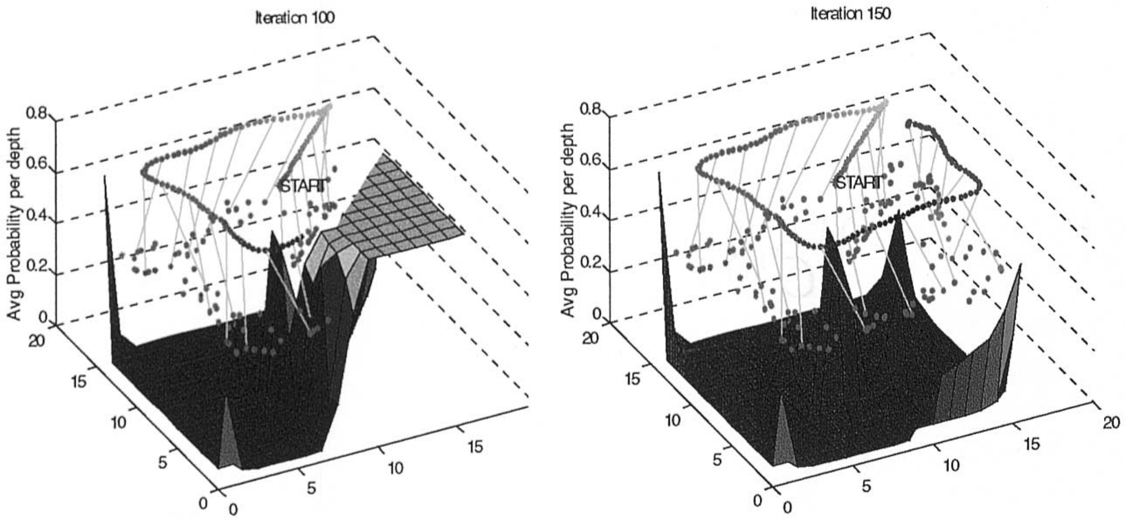

Fig. 11. Resulting path when instinct 3 used.

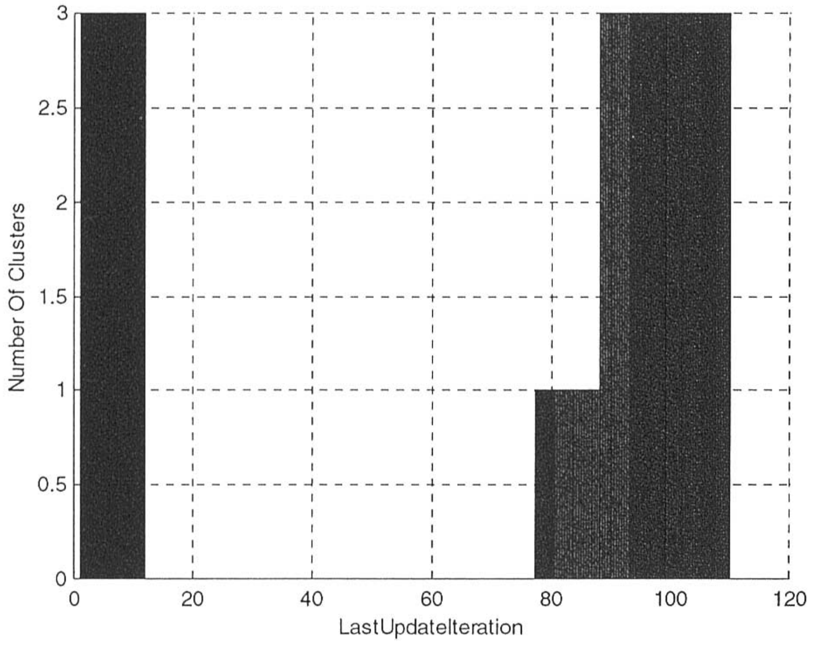

Fig. 12. Clustering results with learning enabled.

ORG starts in position (1000,1000,0) instead of (8000,800,0). The path is different form compared to the path in Fig. 15 but the two paths took approximately the same time (103 iterations versus 111 in Fig. 15) and the final overall probability was near zero in both cases.

X. Conclusion

An unsupervised problem solver implemented as an autonomous agent has been proposed in this paper. Utilizing data objects for flexible knowledge representation and fuzzy logic for storing and retrieving information, the agent provides a new physical symbol system. Abstract initial rules shape the agents behavior and provide the means to discover global intent behind the human supplied heuristics.

By implementing the core learning system using fuzzy logic, the issues of action selection, knowledge representation and attention focusing are addressed with simple and proven means. Fuzzy membership indicators are used to optimize learning and focus the agent's reaction when dealing with large numbers of external stimuli. Sensor modeling based on two different range thresholds merges internal representations of stimuli with the actual sensor data allowing for hybrid model/real world environments. By observing the modifications imposed by the environment on the agents suggested state transitions, the ORG can learn physical constraints and adapt its behavior to comply with the constraints. Speculation and experimentation is used to expand the ORGs abilities and fine-tune its performance. Experimentation allows the ORG to gradually map vague fuzzy reactions to motor functions.

Several simulation results are presented that illustrate typical ORG behavior. Side effects of the instincts and ORG architecture are discussed in detail demonstrating that complex and desirable (for the specific application) behavior emerges out of

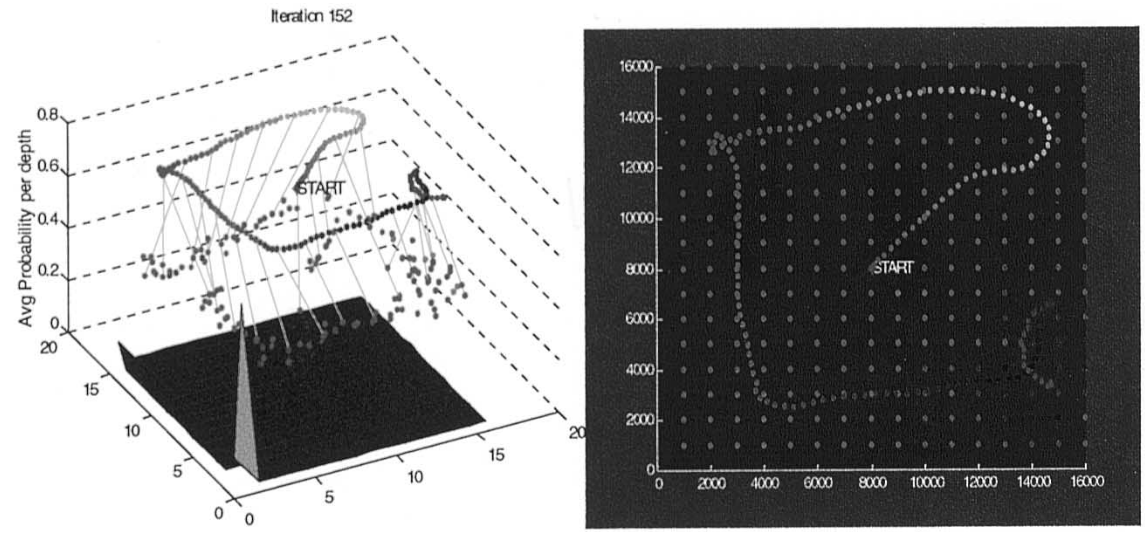

Fig. 13. Results with learning disabled.

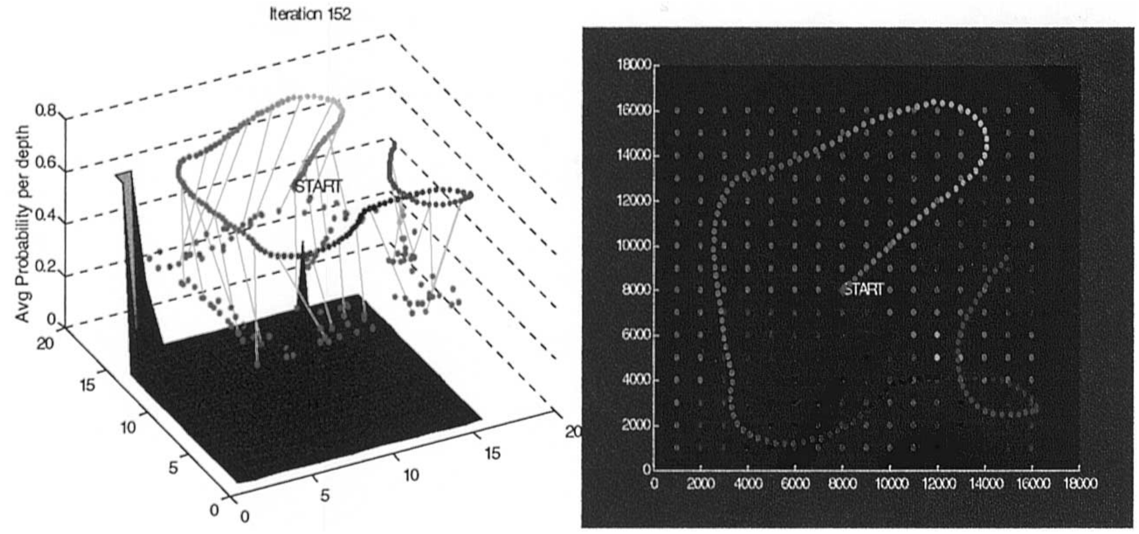

Fig. 14. Results with learning enabled.

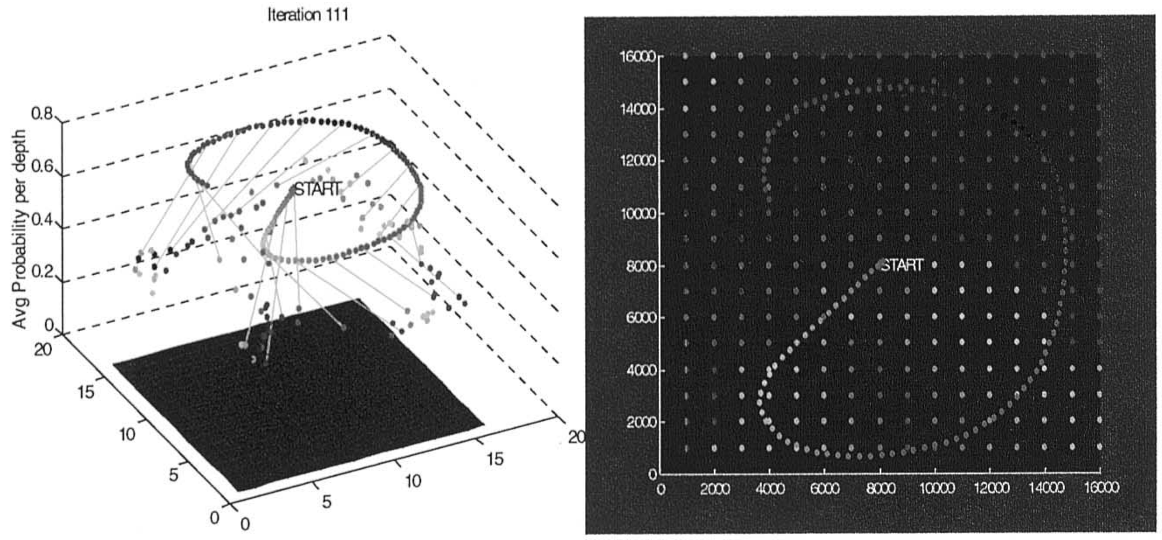

Fig. 15. Learning enabled, larger sensor range.

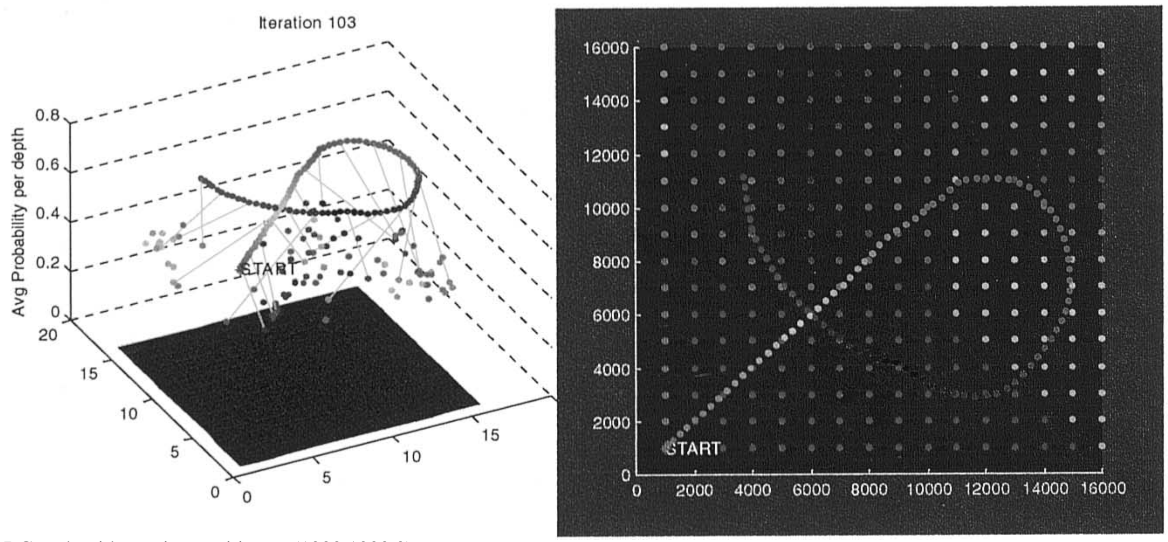

Fig. 16. ORG path with starting position at (1000,1000,0).

simple heuristics and a flexible architecture. The effect of external constraints learning on the simulation results is discussed, demonstrating the subtle and positive influence of learning by self reflection.

APPENDIX A

The following sections describe in detail the procedure used to cluster new associations of condition/effect objects. The terms "cluster centroid" and MEMORY_OBJECT are used interchangeably.

I. STEP 1: INITIALIZE THRESHOLDS

A global *expansion* threshold T_e (1) is used to determine when a new input condition object should become its own cluster. This controls how easily new clusters are created. The threshold is usually initialized to a value close to 1 (e.g. 0.9). A high value makes cluster creation easier since even slightly dissimilar objects will generate a new cluster. The expansion threshold is adjusted each time a new cluster is created or a cluster is replaced. The adjustment is done as follows:

$$T_e = T_e - \left(\alpha T_e e^{-\frac{1}{N}}\right) \tag{1}$$

where $\mathrm{T_e}$ is the expansion threshold, N $=$ max (1, number of existing clusters) and α is the expansion decay. This is set to a small number in the order of 10^{-2}.

A maximum number of clusters can also be optionally set. This number, *MaxClusters*, determines when new clusters can not be added. Instead old clusters get replaced when a new input requires its own cluster. Clusters get replaced using the *relevance* value, R_j for cluster j:

$$R_j = (1-N) + \frac{\mathrm{Iteration}_{\mathrm{Cluster}(j)}}{\mathrm{Iteration}_{\mathrm{Global}}} + \mathrm{MergeCount}_{\mathrm{Cluster}(j)} + E + O$$

where N is the number of active input properties, *Cluster.Iteration* is the iteration this cluster was last modified at and *Cluster.MergeCount* is the number of elements this cluster has. The Boolean variable E *is* set to one if the particular MEMORY_OBJECT has an Effect PROPERTY_OBJECT, zero otherwise. The Boolean variable O is set to one if Cluster.Cnd.Type = InputCnd.Type. The smaller the R_j, the more eligible cluster j becomes for replacement. Justification for this equation was influenced by the task at hand and comes from the following reasoning:

> If a cluster has been used recently it is considered relevant to the current environment. Thus the last update iteration is used. If new associations have been merged several times this cluster, its considered "heavier" and it has a larger "inertia." Thus, it becomes much harder for new knowledge to affect it unless its very similar to it. This justifies the MergeCount variable.

II. STEP 2: CLUSTER NEW MEMORY OBJECT M

When an input `MEMORY_OBJECT`, referred to as M_{input}, is presented to the CORE_LEARN function, the clustering algorithm uses the input condition value matrix ($M_{\mathrm{input}}.Cnd.Values$) to calculate a membership function to each existing cluster condition object.

The input matrix $M_{input}.Cnd.Values$ is represented as matrix U. Matrix U is of size MxK,where

- M is the maximum number of properties that can be concurrently sensed in a single stimulus;
- K is the number of elements in the largest vector property.

The cluster matrix *Cluster(j).Cnd.Values* is represented as $V_j \in R^{Ne}$. Using fuzzy inference the membership value of U_{input} for each cluster $\boldsymbol{j}$ is calculated as follows:

$$m_j = p_j \prod_{k=1}^{N} \left(\prod_{l=1}^{R} \exp\left(- \left(\frac{u_k(l) - v_k(l)}{r_k(l)} \right)^2 \right) \right) \tag{2}$$

where

- $r(j)_k$ is the range for element l of property vector k;
- N is the number of properties common between the Cluster condition and the input condition;

- M is the number of properties in the Cluster condition. Always M >= N;
- p_j is the Partial Information multiplier described in the following.

Note that the cluster condition property vector v_k is used to define the support for the membership function for each input property k.

The crisp output value matrix is generated using

$$O = \frac{\sum_{j=1}^{N} m_j Effect_j}{\sum_{j=1}^{N} m_j}. \tag{3}$$

Effect_j is the matrix formed from the active N property row vectors, from the `EffectObject.PropertyValues` array of cluster j.

Since the clustering algorithm has to account for variable length input vectors, only the same properties active in both the input condition and the cluster condition are compared. When objects of different information content are compared to each other, $M \neq N$, a fuzzy value is calculated to adjust the membership

$$p_j = \exp\left(-\frac{M-N}{M}\right).$$

This equation assigns equal weight to all properties. Optionally the weight can be adjusted by determining the relevance of each observed property.

III. Cluster Expansion/Creation

The decision to merge the new input with an existing cluster or create a new cluster is done as follows.

```
IF $m_{\min} < T_{\text{expansion}}$ AND $N <$ MaxClusters
THEN
  Create new cluster
ELSE IF $m_{\min} < T_{\text{expansion}}$ AND $N >=$ MaxClusters
  Replace cluster $j$, $m_{\min} = m_j$ with new MEMORY_OBJECT $M_{\text{input}}$.
ELSE
  Add new MEMORY_OBJECT $M_{\text{input}}$, to cluster $j$, where $m_{\min} = m_j$.
```

When adding matrix M_{input} to cluster j, *Cluster (j).Cnd* and *Cluster(j).Effect* property value matrices are adjusted to reflect the information content of the new member. The adjustment formula for value matrix U is given as follows:

$$U_{i,\text{new}} = U_{j,\text{old}} - U^{*}_{j,\text{old}} m + U_{j,\text{input}}{}^{*} m$$

where m_j is the combined membership value of M_{input} to cluster j.

References

[1] M. El-Nasr and M. Skubic, "A fuzzy emotional agent for decision making in a mobile robot," in *Proc. IEEE WCCI*, Anchorage, AK, 1998.

[2] M. Wiesmeyer and J. Laird, "Attentional modeling of object identification and search," in *The Soar Papers*, P. Rosenbloom, J. Laird, and A. Newell, Eds. Cambridge, MA: MIT Press, 1992.

[3] Electrical Engineering Dept., Univ. Michigan. A survey of cognitive and agent architectures. [Online]URL:http://ai.eecs.umich.edu/cogarch0/

[4] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE J. Robot. Automat.*, vol. RA-2, pp. 14–23, Mar. 1986.

[5] K. K. Chintalapudi and K. Moshe, "A noise resistant fuzzy C means algorithm for clustering," in *Proc. IEEE WCCI*, Anchorage, AK, 1998.

[6] P. Maes, "Modeling adaptive autonomous agents," in *Artificial Life: An Overview*, C. G. Langton, Ed. Cambridge, MA: MIT Press, 1997.

[7] T. M. Mitchell, *Machine Learning*. New York: WCB/McGraw-Hill, 1997.

[8] H. VanLandingham and G. Chrysanthakopoulos, "Data driven fuzzy logic systems for system modeling," in *Proc. IEEE Systems, Man, Cybernetics*, vol. 1, Vancouver, BC, Canada, 1995, pp. 841–844.

[9] G. Chrysanthakopoulos and R. J. Marks, "Simulated autonomous agents utilized instincts and behavior learning: orgs in Orgland," in *Proc. IEEE Int. Conf. Evolutionary Computation, IEEE World Congr. Computational Intelligence*, Anchorage, AK, 1998, pp. 727–734.

[10] D. C. MacKenzie, J. M. Cameron, and R. C. Arkin, "Tactical mobile robot mission specification and execution," in *Proc. Int. Soc. Optical Engineers*, vol. 3838, 1999, pp. 150–163.

[11] A. Newell, *Unified Theories of Cognition*. Cambridge, MA: Harvard Univ. Press, 1995.

[12] O. Manolov, P. Stanev, S. Noikov, D. Janglova, L. Uher, and S. Vagner, "Intelligent autonomous agent motion control by fuzzy logic in unknown environment," in *Proc. 7th Int. Conf. Artificial Intelligence and Information-Control Systems of Robots*, Smolenice Castle, Slovakia, 1997, pp. 127–132.

[13] J. Bryson, "Cross-paradigm analysis of autonomous agent architecture," *J. Experimental Theoret. Artificial Intell.*, vol. 12, no. 2, pp. 165–189, Apr.–May 2000.

[14] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-3, pp. 28–44, Jan. 1973.

[15] G. J. Klir and T. A. Folger, *Fuzzy Sets, Uncertainty, and Information*. Upper Saddle River, NJ: Prentice-Hall, 1988.

[16] M. desJardins, E. H. Durfee, C. L. Ortiz, Jr., and M. J. Wolverton, "A survey of research in distributed, continual planning," *Artif. Intell. Mag.*, pp. 13–22, Winter 1999.

[17] D. E. Wilkins and K. L. Myers, "A multiagent planning architecture," in *Proc. 1998 Int. Conf. AI Planning Systems*, Pittsburgh, PA, 1998, pp. 154–162.

Georgios Chrysanthakopoulos received the B.S. degree in computer engineering from Virginia Tech Univeristy, Blacksburg, in 1995 and the M.S. and Ph.D. degrees from the University of Washington, Seattle, in 1997 and 2000, respectively.

He joined Microsoft Corporation in early 1996. He is now a Senior Developer on an incubation team exploring distributed operating systems, having worked previously on XBOX System Software Group as the audio and streaming media development lead. Prior to that, he was a development lead in the NT Base Systems Driver group. His research interests include autonomous agents, mobile process algebras, and distributed operating systems.

Warren L. J. Fox (M'89–SM'01) received the B.S., M.S., and Ph.D. degrees in electrical engineering from the University of Washington, Seattle, in 1988, 1990, and 1994, respectively.

He joined the Professional Staff of the Applied Physics Laboratory, University of Washington, in 1988, and earned his graduate degrees as an APL Graduate Fellow. From 1996 to 1998, he was a Scientist in the Mine Countermeasures Group at the NATO SACLANT Undersea Research Centre, La Spezia, Italy. He returned to APL-UW in 1999, where he is currently a Senior Electrical Engineer. He also holds a joint appointment as Affiliate Assistant Professor in the Department of Electrical Engineering at the University of Washington. His research interests are in the areas of statistical signal processing, theoretical and applied ocean acoustics, environmentally adaptive sonar systems, and applications of computational intelligence.

Robert T. Miyamoto received the B.A. degree from the University of California at Irvine in 1973 and advanced to candidacy for the Ph.D. degree in 1976.

He is the Associate Director of the Applied Physics Laboratory at The University of Washington, Seattle, for Applied Research and Technology at the Laboratory. He is responsible for finding new investigative opportunities, bringing together scientists and engineers across departments, and forging communities of applied research to solve problems primarily for the U.S. Navy. He joined the Laboratory in 1979, and has expertise in research and development in oceanography, acoustics, meteorology, signal processing, fisheries, multimedia, and human–systems interaction. From 1993 to 2002, he served as the Head of the Environmental and Information Systems Department. He holds an Affiliate Associate Professorship in the Department of Electrical Engineering at the University of Washington.

Robert J. Marks, II (S'71–M'72–SM'83–F'94) holds the position of Distinguished Professor of Engineering in the Department of Engineering at Baylor University, Waco, TX.

Dr. Marks was awarded the Outstanding Branch Councilor Award by IEEE, was presented with the IEEE Centennial Medal, was named a Distinguished Young Alumnus of Rose-Hulman Institute of Technology and is an inductee into the Texas Tech Electrical Engineering Academy. In 2000, he was awarded the Golden Jubilee Award by the IEEE Circuits and Systems Society. He is also the first recipient of the IEEE Neural Networks Society Meritorious Service Award. He is a Fellow of the Optical Society of America, and was given the honorary title of Charter President of the IEEE Neural Networks Council. He served as an IEEE Distinguished Lecturer. He served a six-year term of the Editor-in-Chief of the IEEE TRANSACTIONS ON NEURAL NETWORKS and as an Associate Editor of the IEEE TRANSACTIONS ON FUZZY SYSTEMS. He was also the Topical Editor for *Optical Signal Processing and Image Science* for the *Journal of the Optical Society on America A*. He served as the Administrative Vice President of the IEEE Circuits and Systems Society. He has over 250 publications, and is the author and coauthor of the books *Introduction to Shannon Sampling and Interpolation Theory* (New York: Springer-Verlag, 1991), and *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks* (Cambridge, MA: MIT Press, 1999). He is a coeditor of five other volumes—all out of print. He is a Founding Member of the University of Washington's Christian Faculty Network, is an Associate Member of Christian Leadership Ministries, and served as the Faculty Advisor to the University of Washington's chapter of Campus Crusade for Christ.

Mohamed A. El-Sharkawi (S'76–M'80–SM'83–F'95) received the B.Sc. degree in electrical engineering in 1971 from Cairo High Institute of Technology, Cairo, Egypt, and the M.A.Sc. and Ph.D. degrees in electrical engineering were from the University of British Columbia, Vancouver, BC, Canada, in 1977 and 1980, respectively.

In 1980, he joined the University of Washington, Seattle, and is presently a Professor of Electrical Engineering. He is the founder of the international conference on the Application of Neural Networks to Power Systems (ANNPS), and a co-founder of the international conference on Intelligent Systems Applications to Power (ISAP). He is a current or past member of the editorial board and associate editor of several journals. He is the author of a textbook on fundamentals of electric drives. He organized and taught several international tutorials on intelligent systems applications, power quality and power systems, and organized and chaired numerous panel and special sessions in IEEE and other international conferences. He has published over 160 papers and book chapters in these areas and holds five licensed patents.

Dr. El-Sharkawi is the Vice President for Technical Activities for the IEEE Neural Networks Society. He is a founding Chairman of several IEEE task forces, working groups, and subcommittees, and is the Editor and Co-Editor of several IEEE tutorial books on the applications of intelligent systems.

Michael Healy (S'68–M'69) received the M.S. degree in mathematics from the University of Idaho, Moscow.

He served as an Associate Technical Fellow, Phantom Works at The Boeing Company. He is currently an Affiliate Associate Professor and Member of the Graduate Faculty in the Department of Electrical Engineering at the University of Washington, Seattle, and a Research Scholar at the Department of Electrical and Computer Engineering at the University of New Mexico, Albuquerque. His research interests include the formal semantics of neural networks and knowledge-based systems. At Boeing, he produced optimization applications in preliminary design, computational geometry, economics, resource allocation, optimal flight control, and structural design. He was responsible for bringing advanced optimization methods and software to the Boeing Mathematical/Statistical Libraries. During his last 12 years at Boeing, he performed research in neural networks and formal methods for software engineering and knowledge based systems development. An outgrowth of his neural network research was the company-wide development and implementation of a neural information retrieval system by software development and engineering personnel.

Dr. Healy is a member of the American Mathematical Society (AMS) and the International Neural Network Society (INNS). He served on the Advisory Board of the Optimization Special Interest Activity Group (SIAG/OPT) for the Society of Industrial and Applied Mathematics (SIAM), and also organized and chaired the first Minisymposium on Optimization and Supercomputing at the SIAM 35th Anniversary Meeting. More recently, he coorganized a session in Neural Networks at Northcon '90 in Seattle, organized and chaired an invited session on Adaptive Resonance Theory neural networks at the 1994 IEEE International Congress on Neural Networks at Orlando, co-organized a special-session block (including a plenary lecture) at the 1996 IEEE Conference on Neural Networks in Washington, DC, and served on the Program Committee for the Algebraic Methodology and Software Technology conference (AMAST 2000 at the University of Iowa). He has also served as an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS.