A Multi-Level Organization of Semantic Primitives for Learning Models of Environment Autonomously from Continuous Data for Design

(*Extended Abstract*)

Sattiraju Prabhakar and Greg Smith
School of Computing Sciences
University of Technology, Sydney
PO Box 123, Broadway, NSW 2007, Australia
Email: {prabhaka, gjsmith}@socs.uts.edu.au

Autonomous Design Agent

In this paper, we present an overview of the project on Autonomous Learning Design Agents which began recently. An agent that addresses the task of *Design by Autonomous Learning* (DAL) builds an abstract model of the environment from the sensory data with the goal to modify the environment to meet a new set of functionalities. Any agent that addresses the DAL task is confronted with a task of transforming the underlying causes that produce the continuous data of the environment to produce another set of continuous data. In our work, we explore the possibility of such a transformation through a hierarchical order of discretizations each of which allows the agent to act differently, and yet allows the agent to draw some global conclusions.

The machine learning research has identified formulation of environment models through multiple levels of abstractions (Pierce and Kuipers 1997). Such a formulation brings to focus some issues that need to be addressed for DAL task.

1. How does an agent recognize satisfaction of a global and abstract goal from continuous data?
2. How can an agent maintain the grounding of discretized model at different levels, in continuous data?
3. How can an agent decide upon local actions based upon global models?
4. How can an agent generate discretized models and yet *open* to new continuous data?
5. How does an agent decide upon environmental transformations that result in new continuous data?

In a theory driven hybrid system, the relationship between the discrete model and the continuous data is supplied by the theory (Zhao 1997). The data driven approaches are able to cope with continuous data in a large number of situations. They are limited to consider abstract global goals at different levels which is often required in design (Chapman 1991). Our approach integrates the strengths of a data-driven and theory-driven approaches and has the major contributing points (Prabhakar 1999).

1. The models, at each level, are built by incorporating the interaction mechanisms of the agent into the continuous data. The interaction mechanisms allow the discretized data to be grounded in continuous data. These models are predictive.
2. Due to such an incorporation, the models can incorporate new continuous data into them.
3. The models are based on the composition of data elements, rather than the concepts of the agent. This adds flexibility to the model to building.
4. The abstraction processes, at different levels, transform the continuous data for prediction about different aspects of the environment - topology, structure, behavior and function.
5. The agent recognizes the goals at a level by identifying a set of patterns in interaction outputs of the agent.
6. The models at each level allow the agent to explore the compositional modeling at that level.

A *design robot* interacts with its environment, learn models of the environment and reorganises the environment such that it delivers a new functionality. An example design robot can design a new drilling rig for its needs in mines, or design a bridge on the surface of Mars. The Mars designer changes its environment on Mars to have the functionality of transporting a vehicle by building a bridge.

Design by autonomous learning is closely related to the functionality of the environment as perceived by the agent. An environment is said to have a functionality if the agent receives a set of percepts for a sequence of actions. An example is the functionality of the environment to produce light where the agent applies pressure on a switch and the agent receives light. In design, the environment initially fails to deliver this functionality. The agent makes changes to the environment structure such that the modified environment delivers the new functionality. The agent also needs to make changes to the environment such that some of its functions are not disturbed. A design problem is specified to the agent as (F_m, F_a), where F_m is the set of functions of environment that need to be maintained, and F_a is the set of functionalities that need to be modified. The set of actions or action sequences A_f, along with the resultant set of percepts P_n, specifies the *design goal.*

All the functions F of the environment may not be known to the agent. Knowing such a complete F may require a very large number of interactions to be performed by the agent. Instead, the agent limits its design goal to the functionalities of the environment it has encountered. The design goal is suggested to the agent, when it encounters these functions in the environment.

A simple example of autonomous learning design task can be illustrated by using the scenario of figure 1. In state1, the agent cannot perceive the environment as having the function of pushing the ball. The agent performs the action specified in the goal - push the stopper. This does not result in the percept that the ball is

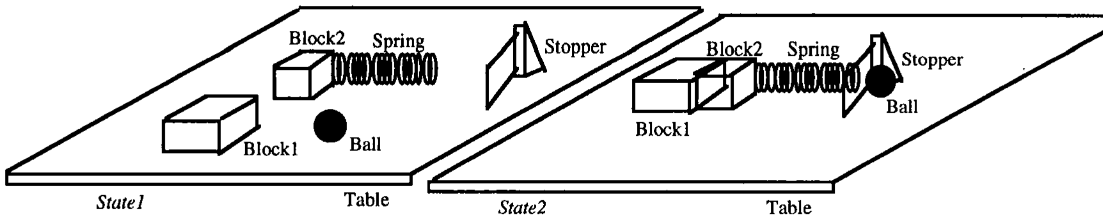


Figure 1. Transformation of an environment that enables satisfy ball pushing function.

in a new position. The agent performs a number of actions on the environment and collects the continuous sensory data. From this data, it builds a discrete model of the environment. This model is predictive and enables the agent to formulate sequences of actions that result in changes to the environment which have desirable perceptual consequences. In state2, the environment is shown as satisfying the perceptual action of pushing the ball.

In this paper, we consider the environments that are closed, accessible and deterministic. We first present a multi-level organization of semantic primitives (MOSP) that allows the incremental discretization of continuous data, then a control strategy that allows the agent to interact with the environment based on the models built and finally we describe the modeling strategy.

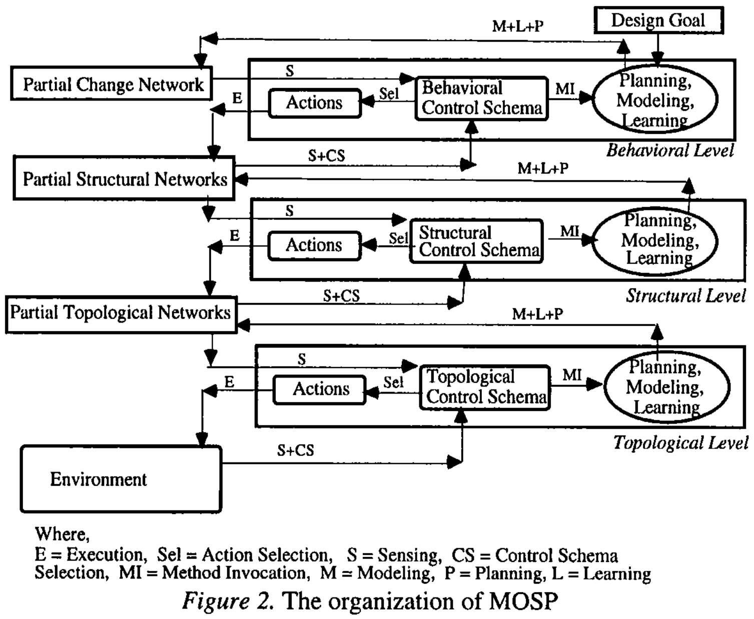
Overview of Methodology

Our multi-level organization of semantic primitives (MOSP) has the following levels - topological, structural, behavioral and functional (see figure 2). As we said earlier, each of these levels corresponds to a level of discretization and a type of pattern over the discretized data for that level. For example, at structural level, the continuous data corresponds to the spatial data and no temporal changes. The patterns are the grouping of the discretized data that correspond to the spatial features of objects.

At each level of MOSP, the semantic primitives include actions and percepts, a set of planning, modeling and learning methods, and control rules. The actions are of two types - perceptual and change. The perceptual actions make the sensor to focus on a new aspect of the environment or continue to focus on the current aspect of the environment. The change actions make changes to the environment.

The control rules enable the agent to respond appropriately to every interaction between the agent and the environment. They invoke actions, and modeling, learning and planning methods. Thus they control the modeling and interaction. The strategy that applies the control rules enable the agent to select te actions such that the agent converges onto the design goals.

Modeling methods map the continuous data of the sensors onto a Situational Action Model. This model helps the agent to select and order the actions at that level. The first step in generating the Situational Action Model is feature discovery at that level from the continuous data of the environment (Prabhakar 1999). For example, at behavioral level, a feature can be a change in the topological relation between two objects. A group of such features form the model that can predict the consequences of the actions.


M+L+P
Design Goal
Partial Change Network
S
E
Actions
Sel
Behavioral Control Schema
MI
Planning, Modeling, Learning
Behavioral Level
S+CS
M+L+P
Partial Structural Networks
S
E
Actions
Sel
Structural Control Schema
MI
Planning, Modeling, Learning
Structural Level
S+CS
M+L+P
Partial Topological Networks
S
E
Actions
Sel
Topological Control Schema
MI
Planning, Modeling, Learning
Topological Level
S+CS
Environment

Where,
E = Execution, Sel = Action Selection, S = Sensing, CS = Control Schema Selection, MI = Method Invocation, M = Modeling, P = Planning, L = Learning

Figure 2. The organization of MOSP

The learning methods at a level generate a Conceptual Model for the actions at that level. Conceptual model is a generalization of the situational action model over a set of interactions considered as examples. This generalization allows the agent to apply its conceptual model to new situations.

Over the concept space generated by learning, the planning generates a sequence of actions that satisfy a set of constraints. These constraints are generated from the design goal or planning at higher levels.

Figure 2 illustrates the MOSP organization. The central aspect of the organization is that the integration between planning, modeling, learning and execution is organized at three levels - at topological, structural and behavioral levels. Functional level is not shown in the figure. Using this organization, the agent forms four levels of egocentric representations, incrementally.

At the lowest level of MOSP, the agent forms a topological interaction representation of the environment. In this representation, the surfaces of the objects provide several topological elements which are connected to each other through perceptual actions of the agent. At the next level, the structural aspects of the environment are represented. At the behavioral level, the changes in the environment are represented as a network of concepts corresponding to the changes. The changes are represented as discrete entities. In the functional level the aspects of the changes in the environment that correspond to the goals of the agent are represented.

The application of methods at each level generates models and generates a feedback to lower levels (this feedback is not shown in the figure 2). This feedback provides goals to be satisfied at lower levels. Thus modeling is both bottom-up and top-down - the models at higher levels are formed from lower level models, while higher level models impose constraints on lower level modeling.

Since central to DAL is the modeling and interaction with the environment, we focus upon these two aspects in he rest of the paper.

Controlling Interaction and Modeling

Two issues are central to dealing with the complexity of autonomous learning task - selecting actions that enable it to achieve the design goal and deciding upon which method to apply. In MOSP, the agent uses a set of control rules called Interaction Control Schemas (ICS) and a simple control strategy.

Initially, the actions at any level of MOSP are not grounded in a model of the environment but on a set of internal parameters of the agent. Hence, the selection of actions is arbitrary, in the beginning stages of interaction. The agent builds a model of the environment after several interactions with the environment. This will enable the agent to select the actions to achieve a specified effect. If the action fails to achieve the required effect, the agent refines its model, thus enabling it to select actions more effectively. This incremental selectivity also applies to the perceptual actions.

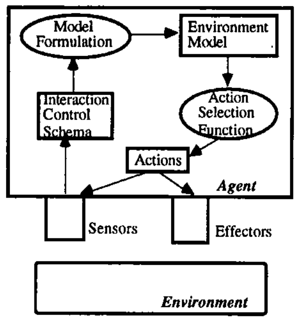


Figure 3. Selection function updating

Figure 3 illustrates this interaction between the agent and the environment. The agent has a selection function at every level of MOSP which helps it to select an action at that level. Initially, this selection function is nil and hence the actions are selected arbitrarily. As the agent interacts with the environment, the environment model is built which improves the selection function and the selectivity of actions within that environment.

The ICS schemas enable the agent to react differently to each interaction situation. The schema that is invoked in a situation sends appropriate signals to methods for planning, modeling, learning, action selection and execution. The agent has a simple forward-chaining strategy of selecting an ICS and executing it. Since the ICS send different signals to different methods, the result can be a complex behavior which converges onto achieving the design goal.

Different levels of MOSP have different perceptual actions. For example, at topological level, a perceptual action is View_Surface, at structural level a perceptual action is View_Object, and at behavioral level it is View_Move. The change actions are also dependent on the MOSP level. An example of behavioral action is Stretch_Hand. An example of navigational action at behavioral level is Turn.

An example of topological ICS for action selection is:

```
IF
    Current-action = Nil
    Current_Percept_Bundle = nonNil
THEN
    Current-action <-- ASelection{At, PBc}
    Current_Percept_Bundle <-- Nil
```

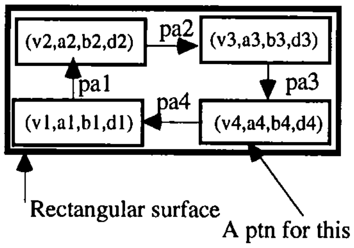
Learning Environment Models

As we said earlier, the agent forms environment interaction models that help it in prediction, and environment modification. The sensory data for the agent is continuous data and can be a spatial distribution of vision data or of a parameter such as magnetic field. Through exploration within this data, the agent forms more abstract models of interaction with the environment. At each level, at every interaction step, first a situational action model is formed and provides a feature space in which the learning of the conceptual model is done by inductive learning. Failure of a concept to predict a percept may invalidate that concept at that level. This may require the concepts or models to be refined at that level or at the lower levels.

Topological Model Formulation

Low level continuous data of environment does not allow comparisons, moving a segment of data to new situations and modification. We need a representation that allows us to do all these operations on the data. Partial Topological Networks (PTN) is such a representation and it provides a basis for passing models to higher levels of MOSP.

At topological level, each surface encountered within the environment is modeled as a group of connected views. A view is a percept received for a spatially bounded two-dimensional region. The representation of this view is called a *topological element* and is represented by a 6-tuple (υ, δ, α, β, θ, ϕ): the view (υ), the distance from the agent (δ), the horizontal angle of surface normal from the direction of viewing (α), the vertical angle of surface normal from the direction of viewing (β), the horizontal beam angle of the view(θ) and the vertical beam angle of the view (ϕ). This 6-tuple also forms the internal parameters of the agent. Topological elements are connected through perceptual or navigational actions taken by the agent. The network of topological elements is called Partial Topological Network (PTN). Figure 4 illustrates a surface of an object and its corresponding PTN.



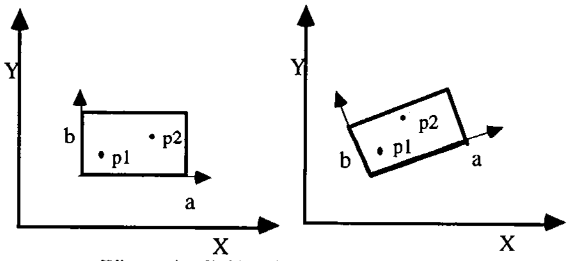
Where,
vi = view, ai = horizontal angle, bi = vertical angle,
di = distance from robot, pai = perceptual action

Figure 4. A rectangular surface and its PTN

At a given time, the agent can maintain several PTNs, each belonging to a different surface. All these PTNs are connected in a bigger network through navigational or perceptual actions.

A view of a topological element can be small or large depending upon the angle of viewing upon the surface. Thus the number of the topological elements is not a measure for the surface area. The perceptual actions determine how much angle or distance the agent needs to move in order to record the view. One of the assumptions for successful generation of the PTNs is that the agent can automatically sense the boundaries of the surface and stop producing the topological elements at the boundary of surface.

Figure 5 illustrates how different points in a view of a topological element become relative to the absolute coordinate system, (X, Y). The view of each topological element has its own coordinate-system (a, b), where a and b are functions of x and y. When a topological element is rotated in an absolute coordinate system (X, Y), its coordinate system (a, b) gets redefined. Every point is automatically gets redefined. This enables the agent to move the views without having to recompute the absolute positions of each point in a view. Further the coordinate systems of the boundaries of adjacent views are connected. Hence, the points within a view become a part of network of topological elements.



Where, pi = f(a,b) and a,b = g(x,y)

Figure 5. Making points of views relative.

If the surface belongs to a three dimensional object, the algorithm is limited to accessing the surface immediately visible.

Structural Model Formulation

The change of environment is often associated with objects rather than with the surfaces. Hence, it is required to generate object representations.

An object representation constitutes of several PTNs. In order to get PTNs for all surfaces of an object, the agent needs to navigate around the object. The other surfaces are visible after navigation. The navigation is triggered by modeling at higher levels of MOSP. For example, a behavioral action called Stretch_Hand can be applied on only objects, thus resulting in their movements. If a single surface PTN is available for an object, the agent actively acquires the rest of the PTNs for the object PTNs. The network of object representation is called a Partial Structural Network (PSN).

The object representation is required to provide its spatial properties, in order for the agent to draw conclusions about the spatial consequences of its actions. The structural level captures object representations along with inter object relationships. The spatial properties are calculated from the vision data that is stored in the views of PTNs.

Behavioral Model Formulation

Usage of behavioral model plays a central role in making changes in the environment. These changes are caused by the agent's actions and the dynamic physical environment. Initially, the agent does not have a behavioral model. The agent explores the environment by arbitrarily selecting and executing behavioral actions, that may result in the movement of the objects within the environment. By observing these changes, the agent is able to build a model of the environment.

The model formed will not only help the agent to select the actions in a specific environment, but also discriminate between two behavioural actions in order to achieve a goal. There are two distinct types of models. Both of these models are derived from the PTNs and PSNs we have discussed earlier. One type of model is attached to individual actions. These models enable the agent to select an action as relevant for a situation by comparing the model with the physical situation. An action can be instantiated many times, each with a different model. The second type of model orders two or more relevant actions for a situation with respect to the goals of the agent. Both of these models are learnt.

The agent develops the behavioral model in two steps. First it forms the situational action model along with a feature space. The features are derived from the changes in the PTNs and PSNs. The feature space is made up of different variations or summaries of the 6-tuples that describe a topological element. This feature space is used to encode the (action, percept) experiences of the agent. An inductive learning algorithm generates concepts, from these encoded experiences, that describe the events within the environment.

The agent may not be able to form complete behavioral models of objects due to incompleteness of PTNs or PSNs. In this case, the agent poses a constraint on lower level modeling to acquire additional PSNs or PTNs. WIn order to present the behavior of the learning algorithm, we first discuss the generation of feature space and then the concept space.

Model Formulation in a Passive Non-Impeding Environment

The passive environment changes only when it is acted upon. In a non-impeding environment, each physical object does not interfere with the changes in the other. An example of such an environment is a single block sitting on a table and no other blocks are present on the table. The movements are unimpeded within the environment of the table. We will explain through an example, how an agent models the environment under such situation. Let us say the agent executes an action that results in the motion of a block. One such behavioural action is Stretch_Hand: $(\delta 1, \alpha 1, \beta 1) \dashrightarrow (\delta 2, \alpha 1, \beta 1)$. This action results in increasing the distance between the agent and the block, but does not change the angles of the agent with respect to the block. The modeling algorithm will summarise the differences between the PTN for the starting position of the block and the PTN for the destination position of the block, as follows.

$\forall t1, t2.\ (t1 \in \mathrm{PTN1} \wedge t2 \in \mathrm{PTN2}) \wedge \mathrm{Stretch\_Hand}(\mathrm{PTN1}) \Rightarrow (\mathrm{distance}(\mathrm{PTN2}) = \mathrm{max\_length}(\mathrm{Hand})) \wedge (\mathrm{distance}(t2) > \mathrm{distance}(t1))$

This model of the changes retains its links with the PTNs. This allows any further generalisation required of model, by making use of the underlying PTNs.

The agent may not be able to sense all the consequences of an action. In that case, the agent may not be able to develop a correct model of the environment. The agent develops a limited model of the environment and keeps modifying it till the functionalities required of the design goal are satisfied.

Model Formulation in a Passive and Impeding Environment

In a passive impeding environment, a behavioral action may not be completed. For example, the stretching of the robot hand to its full length may not be possible due to interference by another block. In this case, modeling the changes requires taking into account the action and the blocking effect. The agent is able to perceive that the action has been impeded by observing the change in the internal parameter called the length of hand. This will prompt the agent to select and execute a few actions that can get the agent the PTN of the blocking object. The comparison between the blocking PTN and the moved block will reveal that they have the same distances. This is summarised in the following model.

$\forall t1, t2.\ (t1 \in \mathrm{PTN1} \wedge t2 \in \mathrm{PTN2}) \wedge (\mathrm{distance}(\mathrm{PTN3}) < \mathrm{distance}(\mathrm{max\_length}(\mathrm{Hand})) \wedge (\mathrm{angles}(\mathrm{PTN1}) = \mathrm{angles}(\mathrm{PTN3})) \wedge \mathrm{Stretch\_Hand}(\mathrm{PTN1}) \Rightarrow (\mathrm{distance}(\mathrm{PTN2}) < \mathrm{max\_length}(\mathrm{Hand})) \wedge (\mathrm{distance}(\mathrm{PTN2}) = \mathrm{distance}(\mathrm{PTN3}) \wedge (\mathrm{distance}(t2) > \mathrm{distance}(t1))$

Model Formulation in an Active and Deterministic Environment

An active environment can change without being acted upon. In this case, the agent needs to do more than act and observe. It needs to actively *track* the environment for its changes. Since the agent does not have the model of the environment, perceptual action selection is not purposeful. That is, the agent cannot decide which aspect of the environment it should sense, how long it should sense, what is the rate at which sense. Our methodology has two aspects:

a. The agent has an initial PTN of an object. The agent will be able to perceive that there are some changes in the environment by repeatedly generating PTN of the current physical situation and comparing in between PTNs. If there are some changes in the object, then it will use the PTN to model the changes. It adapts these changes to perceptual actions. These changes act as models to predict the changes. The failure of prediction leads to modifying the model. This allows the agent to incrementally adapt to track the environmental changes.
b. The second method is for the agent to design controlled experiments and build limited models of the environment which are used to build larger models. This method is not discussed further here.

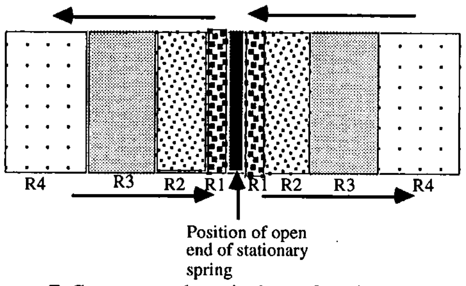


Figure 7. Summary description of spring PTN

To understand the modeling involved in changes of the active environment, consider the spring of figure 1 again. The agent forms a PTN of the spring from one side. Then a behavioral action Sweep_Hand is applied on the open end of the spring. This results in an oscillation of the spring. The agent observes the changes between the new PTNs and the stationary spring PTN. Based on the differences, it forms a model and it keeps observing the differences. The result of these observations is a sequence of PTNs, each slightly different from the others. A summarising algorithm is applied on this sequence of PTNs to generate a statement of the oscillation of the spring. This summary description of the changes in the PTNs captures the oscillations of the spring (as shown in a pictorial form in figure 7). The length of the overall rectangle corresponds to the length of the spring, and the width corresponds to that of the spring. The topological elements are spread throughout the overall rectangle. This rectangle consists of several smaller rectangles each of which corresponds to the density of changes occurred within that region. The rarer the region is, the smaller the changes occurred in the topological elements within that region. For example, the region R4 has smallest amount of changes compared to other regions. This means that the topological elements change less frequently compared to other regions. This frequency distribution is symmetrical. Further, the change in PTNs has a direction as illustrated in figure 7.

Linking Goals to the Action Models

The models we discussed till now help the actions to be selected within a physical situation. But they are not sufficient in informing the agent of the relevancy of an action for a goal compared to the others. The model learning needs to maximise this relevancy. The agent develops a model that is able to compare several actions for their relevance to the goal. At each interactive control situation, the agent compares all the applicable actions, based on their associated models. Depending upon the closeness of their predicted consequences they are ordered. This linking between the goal, the consequences of actions and their attributed priorities forms a central model. This model is revised at each interactive control situation.

Related Work

Our solution for a hybrid system is related to several areas - Machine Learning, Computer Vision, Design and Robotics.

In machine learning, especially in autonomous learning, two approaches are followed prominently to handle continuous data. The first, a hypothesis driven or a theory driven approach uses a prespecified associations between concepts about discrete data and patterns in continuous data (Zhao 1997). The other approach has behavioral mechanisms that interact with continuous data but do not consider the goals at various levels (Chapman 1991). In our approach, we derive associations between concepts and data patterns which are dynamically changed as new associations become necessary.

In computer vision, for recognition of objects from continuous data, associations are used in between schemas and continuous data (Stark and Bowyer 1996). In our case, the initial hypothesis space is set up bottom up from data, then hypothesis driven exploration refines this hypothesis space.

In our work, discovery of several new features at various levels is made (Zhao 1994, Yip 1991, Shen 1993, Fawcett 1993). Design systems need to explore the continuous data while modifying the environment. Often most design systems explore the continuous data without a need to remodel (Smith, Stalker and Lottaz 1996). Our work integrates exploration, modification of the models and changing the environment. Pierce and Kuipers use a multi-level organization for semantic primitives for learning from environment (Pierce and Kuipers 1997). Their approach is statistical, whereas our approach is symbolic.

Discussion

In our methodology, the agent starts with no internal search space. Through interaction with environment, it creates and structures a search space. Search in this space together with exploration in the environment guides the agent to achieve the design goals. A multi-level organization of primitives, by discretizing continuous data at various levels, generates environment with desired continuous data.

A constructive induction algorithm has been implemented for discovering topological change features (Prabhakar 1999). For this algorithm to operate efficiently in MOSP either heuristics or efficient interaction among multiple levels need to be present. This is a very new project and rest of the system is still being implemented to perform simple design tasks in simulated environments.

References

Bailey-Kellogg, C. and Zhao, F. 1997. Spatial Aggregation: Modeling and Controlling physical fields. In *Proceedings of 11th International Workshop on Qualitative Reasoning*, 13 -21. Cortona: Italy: Instituto do Analisi Numerica C. N. R. Pavia.

Chapman, D. 1991. Vision, Instruction, and Action. Cambridge: Mass.: The MIT Press.

Fawcett, T. E. 1993. Feature Discovery for Inductive Concept Learning, COINS Technical Report 90-15, Department of Computer and Information Science, University of Massachusetts.

Pierce, D and Kuipers, B. 1997. Map Learning with Uninterpreted Sensors and Effectors. *Artificial Intelligence* 92: 169 - 227.

Prabhakar, S. 1999. Compositional Constructive Induction: Discovering Topological Features of Environmental Changes from Vision Data. Submitted to AAAI-99 Conference.

Shen, W. 1993. Discovery as Autonomous Learning from the Environment. *Machine Learning* 12, 143 - 165.

Smith, I., Stalker, R. and Lottaz, C. 1996. Creating design objects from cases for interactive spatial composition. In *Proceedings of Artificial Intelligence in Design*, 97 - 116. Dordrecht: Kluwer Academic Publishers.

Stark, L and Bowyer, K. 1996. *Generic Object Recognition using Form and Function.* River Edge: World Scientific.

Yip, K M. 1991. Understanding Complex Dynamics by Visual and Symbolic Reasoning. *Artificial Intelligence* 51, 179 - 221.

Zhao, F. 1994. Extracting and Representing Qualitative Behaviors of Complex Systems in Phase Space. *Artificial Intelligence* 69 (1 - 2), 51 - 92.