Applying Unsupervised Learning and Action Selection to Robot Teleoperation

Mark Witkowski
Department of Electrical and Electronic Engineering
Imperial College of Science, Technology and Medicine
Exhibition Road
London SW7 2BT
m.witkowski@ic.ac.uk

26.3.99

Abstract

Unsupervised learning and supervised remote teleoperator control for robots may seem an unlikely combination. This paper argues that the combination holds advantages for both parties. The operator would like to "instruct" the robot without any special effort, and then be able to hand over some or all of the tasks to be performed without loss of overall supervisory control. In return, the learning algorithm receives a continuous stream of exemplar data relevant to the tasks it might later be asked to perform. We consider an unsupervised learning method, the Dynamic Expectancy Model, and a teleoperator "architecture" offering just such a serendipitous combination.

1 Introduction

This paper describes the Dynamic Expectancy Model (DEM), which combines an unsupervised learning component with a mechanism to create "Policy Maps" for action selection dynamically from this learned information as tasks arise or task priorities change. Action selection and learning models based on Reinforcement Learning ([18], [20], [21], [5] for a broader survey) propagate the effects of occasional reward backward to rank sense-act pairings in the form of a "policy map" according to iteratively developed estimates of future reward. Classifier Systems ([3]) also adopt the notion of propagating the effects of occasional reward, using a "bucket-brigade" method, and combine behavior selection with a genetic algorithm approach to create new classifier selection elements. Action selection may be pre-defined, with no little or no learning component ([9], and [19] for a useful review).

Unsupervised learning is achieved in the Dynamic Expectancy Model by exploiting some interesting properties of prediction. Prediction is used to drive the learning process in two ways. First, the occurrence of unpredicted events allows the system to determine when new predictive rules ("µ-hypotheses") should be formed - *strategic learning*. Subsequent the effectiveness of these µ-hypotheses at making predictions allows the system to determine their quality without reference to any other source of reward or external verification - *tactical learning*. When set a goal the DEM automatically computes and maintains a Dynamic Policy Map (DPM). It may then select actions reactively according to a current perceived situation and a computed "best estimate" of the remaining effort required to achieve the current goal. The DEM develops the schema based or constructivist view ([4], for example) and differs from reinforcement learning and bucket-brigade based methods in two important ways. First, "reinforcement" is generated internally, based on predictions - the system learns in the absence of imposed "reward". Second, the policy map is computed "on-demand" from learned rules, specific to a desired outcome. It is therefore able to adapt its reactive action selection behavior rapidly to changing and widely varying circumstances.

In operation the Dynamic Expectancy Model builds and maintains a number of "list" data-structures. Section 2 defines the three principal lists, the Hypothesis List, the Sign List and the Action List. Sections 3 and 4 introduce predictions and the Prediction List, and describe how prediction is used to create µ-hypotheses and verify them. Sections 5 and 6 introduce goals and the Goal List, and describe the process of setting tasks and building a Dynamic Policy Map, from which actions may be selected to control a robot autonomously. Section 7 illustrates unsupervised learning and policy map construction with a concise example. The remainder of the paper then considers issues of "skill-transfer" ([6]), in the context of the Dynamic Expectancy Model, that arise when a human operator works co-operatively with a robot. The paper further argues that significant advantages will accrue to both operator and learning program as a direct result of this cooperation. Previous research in this area ([2], [7], [8], [12], [15]) demonstrates that

Proc. TIMR 99 "Towards Intelligent Mobile Robots", Bristol 1999.
Technical Report Series, Department of Computer Science, Manchester University, ISSN 1361 - 6161.
Report Number UMCS-99-3-1. http://www.cs.man.ac.uk/cstechrep/titles99.html

skill transfer through learning is both possible and that it can decrease the time required to complete tasks. Although the Dynamic Expectancy Model takes a radically different approach to learning from these examples, many of the problems and issues to be addressed are shared.

A note on notation. Each of the five lists is denoted by a single, upper case, calligraphic letter ($\mathcal{H}$ (Hypothesis), $\mathcal{S}$ (Sign), $\mathcal{A}$ (Action), $\mathcal{P}$ (Prediction) and $\mathcal{G}$ (Goal)). Individual elements are denoted by lower case letters in the same font (h, s, a, p and g respectively). Sub-sets of lists, and individual elements that must be identified across steps or cycles are indicated by superscripts (e.g. $\mathcal{S}^*$). The symbol '$\in$' may be read as "member of", '$\cup$' as "union of" and '$\cap$' as "intersection of". The connective '$\wedge$' is best interpreted as "concurrent with" in this context. Other symbols are explained as they are encountered.

2 µ-Hypotheses, Signs and Actions - Making Predictions

"Knowledge" is accumulated within the Dynamic Expectancy Model as many individual and independent three-part predictions, triples referred to as *µ-hypotheses*. They are so named because each µ-hypothesis is a testable "micro observation" about the robot and its world that can be created, verified and used by the Model. These predictions are encapsulated into context - action - outcome triples. All µ-hypotheses known to the DEM at any time are retained in the *Hypothesis List* (abbreviated to $\mathcal{H}$). The form of the µ-hypothesis (h, $h \in \mathcal{H}$) is:

$$h:\ s' \wedge a \rightarrow^{t} s'' \qquad \text{(eqn. 1)}$$

Each µ-hypothesis encapsulates a "prediction", which can be read as "performing the action a in the context of s' predicts the occurrence of the condition s'' at time *t* in the future." This paper builds on the conjecture that the proper interpretation of this triple is that of a prediction, and that the strength of the connection ("→") should depend only on the predictive performance of the unit. A conventional view would hold that the strength of the connection should be related to any goal or task specific "desirability" of s''. By adopting the predictive view strength changes can be made internally, just by testing whether the predicted event did or did not occur, independently of reward or reliance on an external agent to indicate correctness. This leaves the µ-hypothesis uncommitted to any particular goal; learning is not task dependent. Strength changes can be applied immediately the prediction is verified, and always attributed to the specific µ-hypothesis that made the prediction.

The context (s') and outcome (s'') terms in eqn. 1 are *Signs* drawn from the *Sign List* (abbreviated $\mathcal{S}$, thus $s' \in \mathcal{S}$, $s'' \in \mathcal{S}$). Signs both define and detect *situations* that can be recognised by the Model. Signs are compound items, conjunctions of elemental sensory items ("Tokens"). Thus Signs are derived from, and form the interface to the sensor apparatus available to a physical robot. At each cycle in the execution of the algorithm each Sign is either detected or is absent and so evaluates to *active* (the condition defined by the conjunction of Tokens is met) or *inactive* (the conditions are not met) for that cycle. All active Signs for a cycle are held on the Active Sign List $\mathcal{S}^*$, a subset of $\mathcal{S}$. The first time a Token is encountered the DEM automatically creates a Sign (containing only that Token) and adds it to the Sign List. New Signs are also appended to a working list $\mathcal{S}^{new}$. This list drives the structural learning process. The (structural) learning mechanism in turn provides a mechanism to successively refine predictions available to the robot by adding Tokens to the context Sign conjunction. Tokens, Signs and actions have no *a-priori* meaning to the learning mechanism in the Model, and may be left anonymous or named arbitrarily to suit operator or user.

As a special case individual Tokens (and hence their derived Signs) will equate directly to an observable "State", a unique situation detected reliably by the Sign. A Sign may also equate to a "partially observed state" (P.O. State), a unique situation incompletely detected by the Sign. Equally, a Sign may just represent a collection of sensory conditions available to the robot, from which it must create predictions of varying reliability and repeatability. This notion of "State" is useful when considering robot navigation tasks. The robot wishes to be able to recognise particular locations repeatably and so form reliable prediction based on the (movement) actions it makes. Partially observed locations lead to unreliable predictions about movements. In other situations (skill acquisition, for instance) this notion of "state[1]" is less applicable. In

[1] Even this notion of "State" ignores the temporal component (this place, now). Similarly, adding Tokens to a Sign makes it more restrictive, deleting them less restrictive. It is, of course, exactly these abstractions that make the Sign a useful representation.

these circumstances we expect the formation of many "candidate" μ-hypotheses to be created, verified and discarded while a viable list is being formed.

Actions, (a; $a \in \mathcal{A}$, the *Action List*) define the activities the robot may perform. As Signs defined the interface to the sensory apparatus, actions connect the DEM to physical robot actuators. Selected actions are placed onto the $\mathcal{A}^*$ (active) sub-list. Every action a has associated with it an *action cost*. The action cost measure indicates the relative effort that will be required to complete the action. Action costs may be expressed in any units (such as elapsed time or energy expended) that may be determined and consistently applied across all the elements of $\mathcal{A}$. The action cost measure will be used in the "cost estimation" process for goal directed action selection. The DEM also maintains a memory of recent activations and their associated timings for both Signs (from $\mathcal{S}^*$) and actions (from $\mathcal{A}^*$). Information held on these *activation traces* is used by the structural learning component to construct new μ-hypotheses.

A μ-hypothesis is deemed active (and so placed on $\mathcal{H}^*$) whenever both its context Sign (s') and its action (a) are active simultaneously, ($s' \in \mathcal{S}^* \wedge a \in \mathcal{A}^*$). A new prediction p is created and added to the *Prediction List* $\mathcal{P}$ for every instance of an activated μ-hypothesis. Note in particular that this mechanism is invoked for all μ-hypotheses that meet these criteria and that a prediction records three items. First the identity of the Sign predicted (s'' from the active μ-hypothesis). Second the time (derived from *t*, eqn. 1, and the current time) that the Sign is predicted to occur and third the identity of the μ-hypothesis that made the prediction. Elements of the Prediction List are active (and so placed on $\mathcal{P}^*$) when the time element recorded matches the current time. The presence of active predictions drives the corroboration process.

3 Corroboration - Tactical Learning

The predictive ability for each μ-hypothesis is recorded in a numeric variable, the *corroboration measure* (C_m). For each active prediction, the corroboration measure of the μ-hypothesis responsible for the prediction (whose identity was recorded at the time of making the prediction) is modified according to:

$$C_m = C_m + \alpha(1 - C_m) \qquad \text{(eqn. 2)}$$

where the prediction was successful (that is, when $s^p \in \mathcal{S}^*$, s^p being the Sign recorded at the time of making the prediction), and

$$C_m = C_m - \beta(C_m) \qquad \text{(eqn. 3)}$$

where the prediction was unsuccessful. Active predictions are discarded from $\mathcal{P}$ once this step is complete.

The positive *reinforcement rate*, α ($0 \leq \alpha \leq 1$), defines the rate at which successful predictions will strengthen C_m. Similarly, the *extinction rate*, β ($0 \leq \beta \leq 1$), defines the rate at which C_m will be weakened by failed predictions. Where no prediction was made the value of C_m remains unchanged. Sequences of successful (or unsuccessful) predictions give rise to the familiar negatively accelerating learning curve, the values being normalized such that C_m rises asymptotically toward 1.0 (or falls toward 0.0).

4 μ-Hypothesis Acquisition - Structural Learning

Prediction, or rather the failure to predict an event, drives the structural learning component of the DEM, which is responsible for forming new μ-hypotheses. The opportunity to create new μ-hypotheses is indicated by appearance for the first time of a previously unknown ("novel") Sign or by the appearance of a known but unpredicted ("unexpected") Sign. New, unknown, Signs trigger the *creation by novelty* method, recall that the first occurrence of a previously unknown Sign is recorded in $\mathcal{S}^{new}$ specifically to invoke this learning method. The appearance of an unpredicted, but previously known, Sign invokes the *creation by unexpected event* method. Unexpected Signs are detected by comparing the active Prediction List to the active Sign List and applying the method to the unpredicted residue ($\mathcal{S}^* - (\mathcal{P}^* \cap \mathcal{S}^*)$).

In either method a new μ-hypothesis is constructed from the novel or unpredicted Sign as 's''', and a Sign (s') and action (a) drawn respectively from the recorded activation trace of values in the Sign and Action Lists. The timing relationship (*t* in eqn. 1) is derived from their relative positions in the respective memory traces. Note that the structural learning mechanism is independent of the source of the Signs and actions it will employ. Riolo [16] and Shen [17] have described broadly similar strategies for "rule" creation triggered by "surprise" events.

To limit the rate at which new μ-hypotheses are created the user may specify a *learning probability rate*, λ, which determines the probability with which a new μ-hypothesis will be formed given one of these opportunities to do so. The Dynamic Expectancy Model also defines methods for differentiating partially effective μ-hypotheses (by adding Tokens to the existing context Sign), and removing ineffective ones. The requirements for such additional processes are considered further in [23].

5 The Action Selection Mechanism

At each execution cycle the robot must have some action to perform. Normally, the Dynamic Expectancy Model operates in two distinct modes for action selection (1) *Goal directed selection*, (2) *Exploratory selection*; for the purposes of this paper a third action selection option is introduced, (3) *Guided selection*. Whenever goal directed action selection is selected, the algorithm attempts to construct a *Dynamic Policy Map* (DPM) from which it may select an action. The construction and use of the DPM is described later.

Where no goal is set the system selects actions in exploratory mode. If no exploratory mode is defined, or it is disabled, then the robot will select no actions but wait for guided selection from the operator. In the current implementation, exploratory selection is made on a random basis. Regardless of how the action was selected the learning mechanism continually monitors the activities of the robot, and corroborates existing μ-hypotheses and creates new ones according to the learning strategies described.

Goals - instructions to the system to select actions with respect to some purpose - are held on the *Goal List* ($\mathcal{G}$). Individual goals are drawn from the Sign List. Placing a Sign on the Goal List is a signal to the robot that it should be motivated to select actions that cause that Sign to become activated (i.e. appear on $\mathcal{S}^*$). Each goal Sign on $\mathcal{G}$ is assigned a *priority*. The goal with the highest priority at any time is referred to as the *top-goal*. Once a goal Sign has been activated (i.e. it appears on $\mathcal{S}^*$), it is deemed *satisfied* and removed automatically from the Goal List. The next highest priority goal becomes top-goal, or the Goal List becomes empty.

Purposive goals are, by definition, largely domain specific - they serve some purpose. The Dynamic Expectancy Model provides two distinct routes to setting goals. (1) Goals may be programmed into or be inherent to the robot, or (2) they may be imposed externally by directly manipulating $\mathcal{G}$. The former route equates directly to our intuitive notion of primary reinforcer. Some things, such as food for a hungry animal, or water for a thirsty one, inherently motivate because they are "programmed" to do so. In the mobile robot context the detection of a "battery_low" Sign may cause an "on_charge" Sign to be placed on the Goal List, possibly with a priority related to the extent of battery discharge. The latter route provides an operator with a method with which to manipulate the goal-driven behaviour of the robot directly.

6 Building the Dynamic Policy Map

Whenever a top-goal is available, the DEM will attempt to create a Dynamic Policy Map (DPM) to form a sequence of links from every other Sign in $\mathcal{S}$ to the Sign currently set as top-goal. The DPM is conveniently represented as a graph where context Signs associated with individual μ-hypotheses represent the nodes and actions embedded within individual μ-hypotheses the arcs. The DPM is created by a process of *spreading activation* ([9], [22]) from the top-goal. The method used to construct the DPM is a modified form of the standard breadth-first graph search/construction algorithm ([14]). Each arc has associated with it a *cost estimate*, C_e, value. This cost estimate is computed from the given action cost of $\mathcal{a}$ and the C_m (eqns. 2 and 3) value defined earlier:

$$C_e \leftarrow \text{action\_cost}(\mathcal{a}) / C_m \qquad \text{(eqn. 4)}$$

Consider a situation where C_m is simply p(number of successful predictions|total predictions made) by a μ-hypothesis - the probability that the μ-hypothesis predicts correctly. The cost estimate value C_e is then reasonably interpreted as the total estimated cost for the average number of attempts that must be made with the given μ-hypothesis to achieve the transition. A similar interpretation may be placed on the case for C_m shown in eqn. 4, with the proviso that the "averages" are now biased towards recent experiences.

Each node (Sign) in the graph will acquire a *valence depth*, *v*, indicating the number of arcs, *n*, that must be traversed to reach the top-goal "node". The top-goal has a valence depth of zero, the $\mathcal{s}'$ Sign of any μ-hypothesis that leads directly to the goal (i.e. where $\mathcal{s}''$ = top-goal) a valence depth of 1, and so on. The *policy value*, P_v, of any node $\mathcal{s}'$ at some depth *n* in the DPM is then expressed as a summation of individual cost estimates ($(C_e)^v$) at different valence depths on the least cost path by:

$$P_v(\mathcal{S}') \leftarrow \min \left(\sum_{v=1}^{v=n} (C_e)^v \right) \qquad \text{(eqn. 5)}$$

The policy value for each Sign $\mathcal{S}'$ implicated in the DPM is computed by adding the cost estimate for its transition to the minimum cost of the path to its $\mathcal{S}''$ node. If a lower cost path is encountered the spreading activation is re-activated for that node to minimise path costs at higher valence levels.

Construction of the Dynamic Policy Map is complete when there are no further µ-hypotheses that can be implicated, and no further path cost minimisation can occur. Following construction of the DPM the DEM has an estimate of the total "cost" to attain the top-goal for every $\mathcal{S}'$ implicated in the map. Once DPM construction is complete, the DEM may simply select the action associated with ($\min(P_v(\mathcal{S}' \in \mathcal{S}^*)$) in the µ-hypothesis containing that $\mathcal{S}'$ and route it to the robot.

If a currently active Sign is included as a node in the DPM (DPM $\cap\ \mathcal{S}^*$), the action α included in the µ-hypothesis arc associated with the Sign node with the lowest P_v (eqn. 5) is selected. This is the first action of a potentially long sequence of actions with the lowest overall estimated cost to achieve the top-goal. Where there is no intersection between the set of active Signs and nodes on the DPM, an exploratory action is selected, or the robot may wait for guidance from the operator. These new actions will either (1) reach the goal directly, (2) lead to a situation where a action selection from the DPM may continue, or (3) cause new µ-hypotheses to be created, which in turn expands scope of the DPM. The DPM is recomputed frequently, whenever goals change, new µ-hypotheses are formed or existing ones have undergone sufficient corroboration to indicate that a different solution path may be preferable.

The DPM is not a plan; it is a mapping between sensory conditions and "best" (as defined by the sequence of lowest policy values) action to take in each case. It does not define a path from start to finish; rather it is a characterisation of the Signs known to the system according to an estimated cost from that Sign to the top-goal. In use it has the form of a reactive look-up table, "if this is the current situation, then select this action". In this respect, it is similar to the policy map constructed by some classes of reinforcement learning algorithms, notably those using *Q*-learning ([18], [20], [21]). The DEM is profoundly different in that the policy is computed (and re-computed) frequently and quickly, relative to a specific goal, rather than as an iteratively formed policy estimating future discounted and anonymous rewards.

While not a "plan" in the conventional sense, it is possible to construct a *policy path* from the DPM, a summary of the "best estimate" path through the graph from the current situation to the desired goal. A policy path appears rather similar to a "plan", indicating the actions that would be taken and the "estimated cost" to reach the top-goal at each step.

7 An Example

To illustrate the two essential of the properties of the DEM algorithm, unsupervised learning and policy map generation, figure 1 shows a simulated robot learning task for navigation. The robot may recognise some 74 individual locations on a grid within the environment. These equate directly to individual Signs (and, in this instance, to "states"). The robot is supplied with actions to traverse between locations. These experimental conditions, but not the actual layout, accurately reflect those used by Sutton [18]. We note that, in simulation, each action takes, on average, 2.66 seconds. This is used as the action cost. Initially the robot is allowed 2000 exploratory (randomly selected) actions. No goal is set nor any other form of reward provided during this period, nevertheless a Sign List and a corpus of µ-hypotheses is constructed in the Hypothesis List by the novelty and unexpected event methods, and subsequently corroborated. Random exploration is inefficient in this environment, the robot tending to become "trapped" in "rooms" for extended periods, but this number of actions ensures that every location is visited more than once. In a situation with operator intervention the robot may be guided around the environment dramatically reducing the overall exploration time.

Immediately following this period of exploration the location "A" is established as top-goal. The DEM computes the Dynamic Policy Map visualised in Figure 1a. Column heights indicate the Policy Value (the minimum total cost estimate) from that Sign/location (their position) to the current goal Sign/location. The graphs do not indicate the action associated with each Sign. The action is selected from the policy map according to the minimum cost path. After so much exploration, the task is learnt well. The Policy Values shown in the Dynamic Policy Map correspond closely to our intuition of the "cost gradient", flowing from “room” edges, through “doors”, along the central “corridor” etc. In most robotic applications there is no clear mapping of cost estimate to place and satisfactory visualisation is harder to achieve (but the policy path provides a partial indication). Next location “B” is made top-goal, and the DPM immediately adopts the Policy Value configuration of Figure 1b. Figure 1c shows the DPM Policy Values for location “C”. The spreading activation algorithm described here is fast, the DPM in these examples being computed in “real-time” (<10mS on a 166MHz Pentium P5 running Linux). Witkowski [23] describes an extensive series of investigations using the DEM under a variety of learning conditions, imposed noise and varying environments.

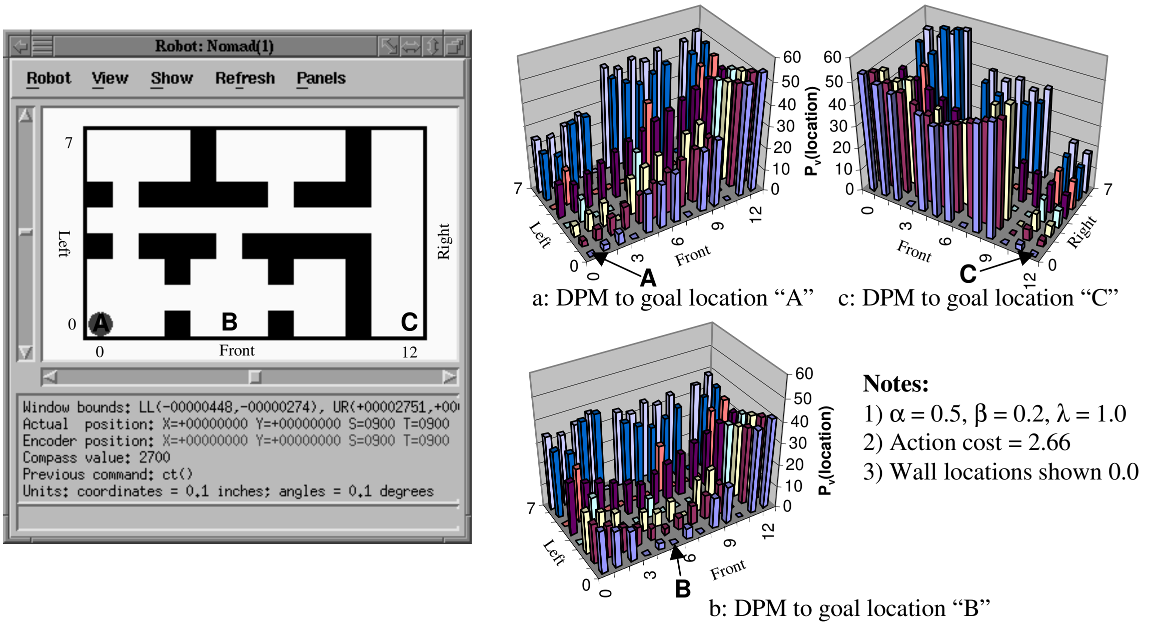

Figure 1: Robot Task and Dynamic Policy Maps for Different Goal Locations

8 Applying the DEM to Robot Teleoperation

Robot teleoperation is used extensively in the nuclear and chemical industries where the operating environment poses direct physical danger, and in offshore and pipeline industries and space exploration where human access is highly restricted, particularly hazardous or acutely expensive ([11], [24]). Teleoperation is finding application in humanitarian tasks such as bomb disposal and mine clearance with their clear dangers ([13]). Increasingly it is being considered for applications in the agricultural, security and construction industries where, although not overtly hazardous, socio-economic arguments can be made in favour of the adoption of teleoperation.

Early teleoperator robots provided direct coupling between human operator and robot with direct visual feedback, through a hazard resistant viewing window or via CCTV images. Increasingly computers and additional instrumentation have been incorporated to provide a more natural control interface to reduce the operator stress and difficulty associated with managing so many degrees of freedom and so greatly increase productivity and effectiveness. Additional problems arise as the distance between operator and robot increases, and where the robot is mobile. In part, these relate to the provision of communications between the operator and the robot. Clearly where a cable or fibre optic connection may be maintained between base station and robot high data rates may be maintained reliably. In some important applications, where remotely operated mobile vehicles must be sent into emergency situations the cable itself becomes a serious liability. In these situations the cable may become entangled on obstacles or severed. Alternative communications methods, such as radio data links, give rise to other problems, including restricted bandwidth and tendency to signal dropout or loss due to signal path occlusion. Particularly demanding applications, such as remote space exploration, highlight all these problems, restricted bandwidth due to strict power constraints, priority to mission (rather than control) data, tendency to data interruption and, additionally, long delays in the “control loop”.

It may seem incongruous to attempt to apply an ostensibly unsupervised learning method to address problems of supervised robot operation, but the combination brings advantages to both. For the operator the robot may eventually be allowed to recommend actions appropriate to the tasks being performed, thereby sharing and so reducing operator effort loading. Under certain circumstances (such as long communications delays), it may further be appropriate to communicate with the robot in terms of task "goals", rather than individual actions. Where communications is interrupted or lost, the robot might be tasked to continue with its mission goals unsupervised. In turn the learning algorithm is presented with a stream of exemplar Signs and Actions which are relevant to the tasks it will subsequently perform. At various time the operator may also pick certain Signs to act as goals. Only when the operator elects to assert those goals will the Dynamic Expectancy Model create a Dynamic Policy Map and produce candidate actions for the robot.

As Barnes and Counsell [1] have pointed out "it can be very disconcerting for an operator to realise suddenly that a robot is apparently ignoring his instructions and instead executing its own set of commands!" To avoid the disadvantages of inappropriate or unexpected lapses into autonomy we exploit some properties of the DEM:

1) Learning of μ-hypotheses and goal-directed action selection generation are independent (though inter-related) processes.
2) Exploratory selection behaviours can be inhibited.
3) Both tactical and strategic learning takes place continuously regardless of the source of Signs and Actions.
4) Unless a top-goal is explicitly established by the operator (or mission goals are enabled) the algorithm will not create a Dynamic Policy Map and so be able to propose actions.

Figure 2 shows an architecture for shared control. To the right is the remote vehicle, with local computer resource to: (1) input sensor information and pre-process it into a stream of Signs, (2) accept the action stream and present it to the actuators and (3) run the DEM controller. To the right is a representation of the operator workstation. It has apparently standard components, screens to present sensor information to the operator and a control panel allowing the operator to select robot actions. Between them are shown a number of data paths, which may be subject to bandwidth restriction and/or delay. Note the arrangements made for sensing and control information to be distributed and shared between robot, DEM controller and the operator. The sensor stream is split, following its conversion at the Sign level, one copy of the stream ('S2') being sent to the DEM algorithm and one ('S1') to the operator station.

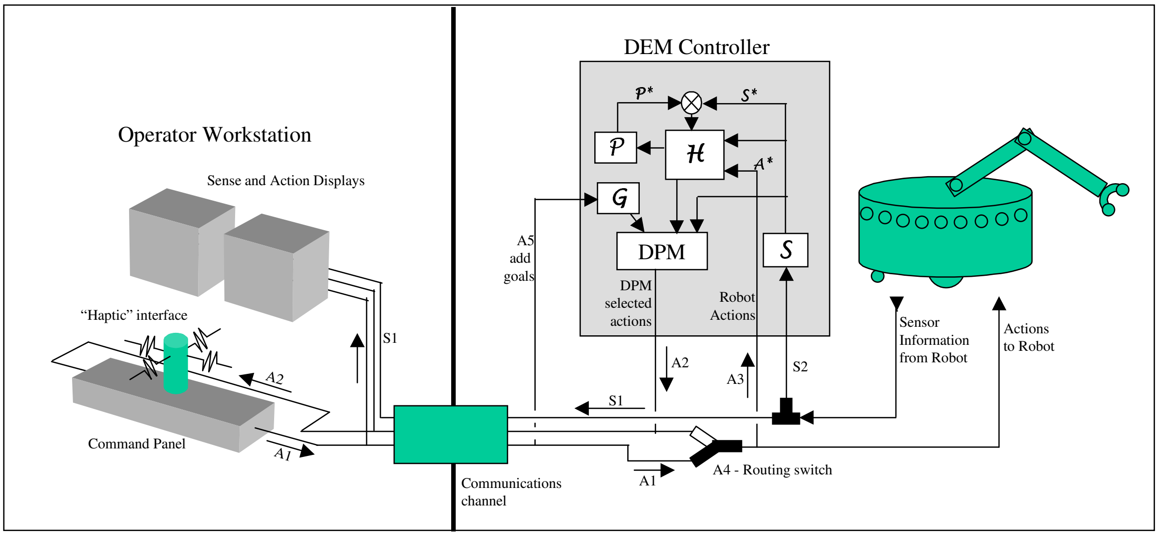

Figure 2: Architecture for Shared Control

The action command pathway is more interesting. Path 'A1' is the normal connection from operator and robot. Path 'A2' routes the action output of the DEM controller to the operator workstation, allowing the operator to intercept and, if necessary, override any actions initiated by the controller. The path 'A3' feeds the actual command sent to the robot into the DEM controller. This ensures that the learning mechanism uses actual commands in order to update the correct μ-hypotheses and learn valid predictions. When the DEM controller is acting normally ("stand alone") the selected action would be routed directly to the learning component). A "routing switch" at 'A4' allows actions selected by the controller to be sent directly to the robot. The operator may activate this switch if autonomous activity is required, or it may be closed automatically if the communications link becomes inoperative. In this latter case, it is appropriate to ensure the Goal List was pre-loaded with mission goals to initiate construction of the DPM and selection of purposive actions. Goal setting is treated as a type of action that the operator can make[2]. Goals are extracted from the action stream and directed to the Goal List in the DEM controller (path 'A5').

The action stream path represented by 'A1' and 'A2' throws up a particularly intriguing application for programmable "haptic" (force reflecting) input devices, such as joysticks[3]. Conventionally these are used to provide the operator with a sensation of the force or resistance encountered at the robotic effector. The possibility presented here is to use the force component to indicate "intention", rather than physical feedback. Normally the joystick is used to select actions from a screen of options. Attempts to select invalid actions can be programmed to present considerable resistance, attempts to select potentially problematic actions signaled by programming an imposed vibration on the grip as the option is approached. When the DEM controller is selecting actions, the operator may monitor the actions proposed both on the screens and by programmed movement of the joystick. However, if events appear to be taking an unsatisfactory turn he can apply a little extra force to overcome the resistance and so override the action. Clearly there are a number of HCI related issues to be considered, as well as how genuine force feedback information may be merged with the "intention" feedback. It also seems likely that this option will only be suited to terrestrial applications or "training" sessions where the time delay component is minimal.

In operation, then, the DEM controller is started, the various Lists empty. With exploratory action selection disabled and no goals available to the controller, the robot just awaits operator action selection. Whenever the operator selects an action a stream of action and Sign change events enters the DEM controller. Newly encountered Signs and actions are registered into their respective Lists. Novel or unpredicted Signs trigger the μ-hypothesis creation methods, and the Hypothesis List is established. Whenever known "context-action" situations are encountered individual μ-hypotheses are activated, make their predictions and have their confidence measures strengthened or weakened according to the eventual outcome through tactical learning. So far, the operator has only been aware of a robot that reacts to his commands. The DEM controller has, during this time, been both acquiring and corroborating a substantial List of μ-hypotheses.

By going about his everyday business remotely controlling the robot, the operator has focussed the learning of the DEM into those areas relevant to the tasks the robot may later be called on to perform autonomously. In contrast, allowing the controller to perform an exploration of such a large search space by random exploration (the default exploratory strategy) invites a potentially catastrophic combinatorial problem. It also sidesteps a host of safety problems related to a robot making strange and unpredictable actions. Strategies have been developed that can significantly improve exploration performance in large feature spaces. Prioritized sweeping [10], for example, but these really represent no substitute for the advantages of guided exploration afforded by operator control.

At any point the operator may elect to designate any of the available Signs as a goal. This will cause the DEM controller to form a Dynamic Policy Map, and, if at least one of the currently active Signs intersects the policy map, it will recommend an action. If the controller can make no recommendation the operator continues to direct the robot towards the desired goal until DPM and S^* overlap. The operator can override the controller's choice or allow it to continue. In deciding whether to intervene the operator can inspect the "policy path" to the goal and judge if it seems plausible. By not intervening at all the operator in effect hands over the robot to autonomous control and need only arrange to monitor and replenish the Goal List from time to time.

9 Summary

This paper has looked at the Dynamic Expectancy Model, which combines an unsupervised learning algorithm with a method for reactive action selection when presented with specific task goals. We then propose a "DEM controller" for robot control that works in cooperation with a human operator in a teleoperation environment and suggest that the combination brings advantages to both. We are concerned that that the operator should suffer no "nasty surprises" with the robot making unexpected actions

[2] This is convenient here, but there are more esoteric reasons why goal setting should be treated as a type of action.
[3] Until recently these were expensive, high precision items. Now reasonably engineered consumer units are cheap and readily available as accessories for the computer games market, and interfaced via a standard MIDI port.

autonomously. In the first stage, then, the operator sees and uses a purely passive robot; but the robot is observing each action and its outcome and making predictions internally. In the second stage, the operator may share his task goals with the DEM controller, and the robot will propose candidate actions from the Dynamic Policy Map formed. The operator may monitor these and override them at will. In the third stage the operator may communicate to the robot in terms of task goals, and permit fully autonomous operation.

References

[1] Barnes and Counsell (1998) "Haptic Communication for Telerobotic Applications", Proc. 29th Int. Symp. on Robotics, "Advanced Robotics: Beyond 2000", N.E.C., UK

[2] Bratko, I., Urbancic, T. and Sammut, C. (1995) "Behavioural Cloning: Phenomena, Results, and Problems", Proc. 5th IFAC Symposium on Automated Systems Based on Human Skill, pp. 143-149

[3]Booker, L.B., Goldberg, D.E. and Holland, J.H (1990) "Classifier Systems and Genetic Algorithms", in: Carbonell, J.G. (Ed.) Machine Learning: Paradigms and Methods, The MIT Press, pp. 235-282

[4] Drescher, G.L. (1991) "Made-up Minds: A Constructivist Approach to Artificial Intelligence", The MIT Press, Cambridge, MA

[5] Kaelbling, L.P., Littman, M.L. and Moore, A.W. (1996) "Reinforcement Learning: A Survey", Journal of Artificial Intelligence Research, **4**, pp. 237-285

[6] Kaiser, M. (1997) "Transfer of Elementary Skills via Human-Robot Interaction", Adaptive Behavior, **5**-3/4, pp. 249-280

[7] Lin, L-J (1991) "Programming Robots Using Reinforcement Learning and Teaching", in: Proceedings of the American Association for Artificial Intelligence (AAAI-91), pp. 781-786

[8] Maclin, R. and Shavlik, J.W. (1996) "Creating Advice-taking Reinforcement Learners", Machine Learning, **22**, pp. 251-282

[9] Maes, P. (1991) A Bottom-up Mechanism for Behavior Selection in an Artificial Creature, 1st Int. Conf. on Simulation of Adaptive Behavior 238-246

[10] Moore, A.W. and Atkeson, C.G. (1993) "Prioritized Sweeping: Reinforcement Learning with Less Data and Less Time", Machine Learning, **13**, pp. 103-130

[11] Nasa (1998) "Nasa Space Telerobotics Program", http://ranier.oact.hq.nasa.gov/telerobotics.html

[12] Nehmzow, U. and McGonigle, B. (1994) "Achieving Rapid Adaptions in Robots by Means of External Tuition", Proc. 3rd Int. Conf. on Simulation of Adaptive Behavior, pp. 301-308

[13] Nicoud, J-D. (1997) "Vehicles and Robots for Humanitarian Demining", Industrial Robot, **24**-2, pp. 164-168

[14] Nilsson, N.J. (1980) "Principles of Artificial Intelligence", N.Y.: Springer-Verlag

[15] Pomerleau, D.A. (1991) "Efficient Training of Artificial Neural Networks for Autonomous Navigation", Neural Computation, **3**, pp. 88-97

[16] Riolo, R.L (1991) "Lookahead Planning and Latent Learning in a Classifier System", 1st Int. Conf. on Simulation of Adaptive Behavior, pp. 316-326

[17] Shen, W-M. (1994) Autonomous Learning from the Environment, Computer Science Press, New York

[18] Sutton, R.S. (1990) "Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming", in Proc. 7th Int. Conf. on Machine Learning, pp. 216-224

[19] Tyrrell, T. (1993) "Computational Mechanisms for Action Selection", University of Edinburgh, (Ph.D. thesis)

[20] Watkins, C.J.C.H. (1989) "Learning from Delayed Rewards", King's College, Cambridge University (Ph.D. thesis)

[21] Watkins, C.J.C.H. and Dayan, P. (1992) "Technical Note: *Q*-learning", Machine Learning, Vol. 8, pp. 279-292

[22] Witkowski, C.M. (1983) "A Parallel Processor Algorithm for Robot Route Planning", Proc. 8th IJCAI, pp. 827-829

[23] Witkowski, C.M. (1997) "Schemes for Learning and Behaviour: A New Expectancy Model", Ph.D. Thesis, Department of Computer Science, Queen Mary Westfield College, University of London, February 1997, 257pp

[24] Woodshole (1998) Woodshole Deep Submergence Laboratory (1998) "WHOI DSL Subsea Vehicles", http://www.dsl.whoi.edu/DSL/DSLvehicles.html