Performance of Lookahead Control Policies in the Face of Abstractions and Approximations

Ilya Levner, Vadim Bulitko, Omid Madani, and Russell Greiner

University of Alberta, Edmonton AB, Canada T6G 2E8,
{ilya|bulitko|madani|greiner}@cs.ualberta.ca

Abstract. This paper explores the formulation of image interpretation as a Markov Decision Process (MDP) problem, highlighting the important assumptions in the MDP formulation. Furthermore state abstraction, value function and action approximations as well as lookahead search are presented as necessary solution methodologies. We view the task of image interpretation as a dynamic control problem where the optimal vision operator is selected responsively based on the problem solving state at hand. The control policy, therefore, maps problem-solving states to operators in an attempt to minimize the total problem-solving time while reliably interpreting the image. Real world domains, like that of image interpretation, usually have incredibly large state spaces which require methods of abstraction in order to be manageable by today's information processing systems. In addition an optimal value function (V^*) used to evaluate state quality is also generally unavailable requiring approximations to be used in conjunction with state abstraction. Therefore, the performance of the system is directly related to the types of abstractions and approximations present.

1 Introduction

Image interpretation is a complex and adaptive task. It is complex in that there is rarely a one-step mapping from image data to scene interpretation; instead, a series of transformations is required to bridge the gap between sensory input and scene description. Examples of these transformations include region segmentation and labelling, texture filtering, and the construction of 3D depth maps, each having a multitude of parameter settings. The task is adaptive in the sense that there is no fixed sequence of actions that will recognize all objects in all settings. For example, the steps required to locate and identify trees in winter are different from the steps required to find trees in summer. In general, the most effective action sequence may depend on many features of a specific image, including lighting conditions, point of view, and the type, shape, and relative position of objects present. We therefore view this task as a dynamic control problem where the optimal vision operator is selected based on the problem solving state at hand. Thus the control policy maps problem-solving states to operators in an attempt to minimize the total problem-solving time while reliably interpreting the image.

S. Koenig and R. Holte (Eds.): SARA 2002, LNAI 2371, pp. 299–307, 2002.

This study is motivated by the Forest Inventory Mapping System (FIMS) [Bulitko et al. 2000] currently under development. Having detailed inventories of forest resources is of tremendous importance to forest industries, governments, and researchers. Such a system would aid planning wood logging (planting and cutting), surveying against illegal activities, and ecosystem and wildlife research. Given the dynamic nature of forest evolution, the task of forest mapping is a continuous undertaking, with the objective of re-mapping the estimated 344 million hectares of Canadian forests on a 10-20 year cycle. Remote-sensing based approaches appear to be the only feasible solution to inventorizing the estimated 10^{11} trees. Despite numerous previous attempts [Gougeon 1993, Larsen, Rudemo 1997,Pollock 1994], no robust forest mapping system exists to date.

Recent research [Draper et al. 2000] has shown that adaptive control policies computed by assuming an underlying Markov decision process (MDP) can outperform hand coded control mechanisms on real-world computer vision tasks. The MDP framework allows for many potential benefits in addressing the control problems in image interpretation effectively, as this framework permits general control models and has a number of successful solution methodologies. In this paper, we present the FIMS system (section 2), which treats object recognition as a Markov decision process (MDP) control task. Furthermore, we introduce state abstraction and action approximation and demonstrate their benefits and shortcomings. Section 3 presents the test bed domain used in our experiments. Finally a lookahead control policy is described and analyzed that operates within the abstracted MDP space to enable efficient application of vision operators within the FIMS domain (section 4).

2 Image Interpretation in the FIMS Domain

In the complete FIMS system, image interpretation is composed of several layers of operators (Figure 1). [Bulitko et al. 2000] explains the benefits of a multi-layer processing approach in meeting the challenges of the forestry domain. At each stage or layer, there are two or more competing operators: several feature extractors, segmenters, labellers, canopy groupers, reconstructors, and renderers. The final labelling contains information such as the location, type, size, age, and other attributes for each tree identified within the image. The system is then able to perform 3D-reconstruction and rendering, enabling FIMS to compute the similarity between the initial image and the rendered image. This allows the intermediate layers as well as the final scene interpretation to be evaluated on their accuracy of interpretation. In each step the control policy (agent) chooses among applicable operators by *selectively* envisioning their effects several plies ahead. It then evaluates the states forecasted using its *heuristic value function,* $\tilde{V}^*$ and selects the operator leading to the most promising state

In the FIMS system, a *sequence* of operators is required in order to produce a 3D interpretation from raw images, and they have to be selected *dynamically* based on the state observed. By viewing the operators as actions and their respective data inputs and outputs as states we can formulate the task of image

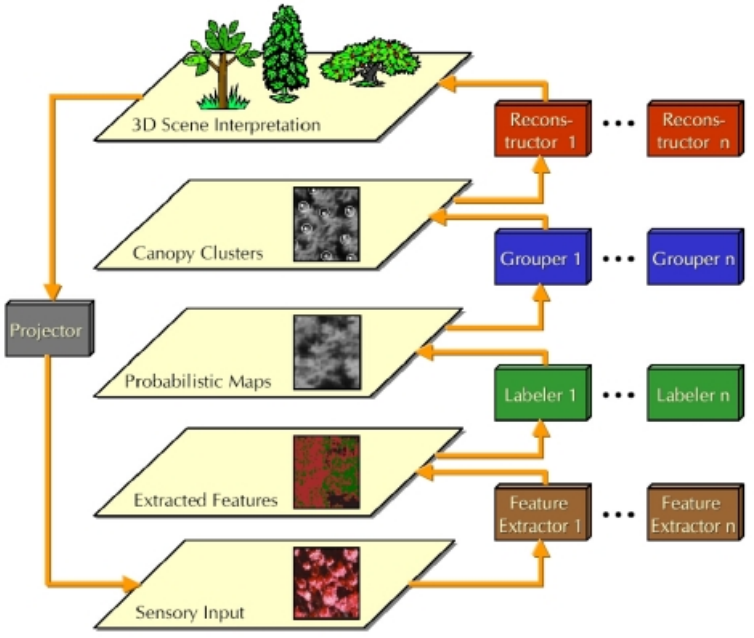

Fig. 1. The layers (phases) of image interpretation in FIMS.

interpretation as a Markov Decision Process [Sutton, Barto 2000]. Therefore, FIMS can be viewed as an extension of the classical MDP framework along the following directions: (i) feature functions for state space reduction, (ii) approximation of value function ($\widetilde{V}^*$) and domain model ($\widetilde{\delta}$) via machine learning methods, and (iii) explicit accuracy/efficiency tradeoff for performance evaluation.

2.1 Indeterminate Goal States

In FIMS, the goal state contains a geo-referenced 3D forest map with the minimum possible amount of error. Unfortunately, such a state is impossible to recognize since the sought forest maps are typically unknown. The lack of recognizable goal states means that we cannot use standard goal-searching algorithms such as A*, IDA*, RBFS, etc. [Korf 1990]. As mentioned previously, the renderer enables the 3D scene description to be transformed back into pixel level data thereby allowing for comparison between the input and final output. Currently this is the only way to determine how close the derived image interpretation matches the underlying pixel data.

2.2 Partial State Observability

Raw states are often enormous in size – FIMS requires on the order of 10^7 bytes to describe a *single* problem-solving state and furthermore the state space is composed of billions of these megabyte sized states. Thus, the raw state space is infeasible to handle. A common remedy is to employ a *state abstraction function* $\mathcal{F}$ that maps large raw states to smaller abstracted states specified via extracted feature values. ADORE [Draper et al. 2000] uses feature functions to reduce multi-megabyte raw images to a handful of real-valued parameters such as the average image brightness or edge curvature. As a result, the control policy operates over the abstracted state space usually recasting the entire problem as a Partially Observable Markov Decision Process (POMDP) [Sutton, Barto 2000].

2.3 Value Function Inaccuracies

In order to help select an operator to apply we use a value function that maps each problem-solving state to a numeric score, thereby setting a preference relation over the set of reachable states. It is typically defined as the reward the system would get from that state on by acting optimally. If the control policy had access to the actual optimal value function V^*, [Sutton, Barto 2000] it would be able to act optimally in a simple greedy fashion, i.e. take actions leading to states (s) with the highest expected $V^*(s)$ score. Unfortunately, as the true value function is usually unavailable, we use an approximation $\widetilde{V}^*$ (denoted with a tilde). Various inaccuracies in the approximate V^* often lead to sub-optimal action choices and force back-tracking, as noted in [Draper et al. 2000].

Two main sources of inaccuracies are present in the $\widetilde{V}^*$. The first source is state abstraction via the use of feature extraction function $\mathcal{F}(s)$, which can map several distinct states to one abstracted state or tile. Formally, there may be distinct states s_i, s_j within our state space S, such that $s_i, s_j \in S$ & $s_i \neq s_j$ & $\mathcal{F}(s_i) = \mathcal{F}(s_j)$. Therefore, $\widetilde{V}^*(\mathcal{F}(s_i)) = \widetilde{V}^*(\mathcal{F}(s_j))$. The second source of errors is noise due to machine learning. Since the optimal value function is being approximated via machine learning techniques, errors are inevitable. Even if the feature function did not combine two distinct raw states into one abstracted state, due to machine learning the two distinct abstracted states may still be mapped to the same value, ie $\widetilde{V}^*(\mathcal{F}(s_i)) = \widetilde{V}^*(\mathcal{F}(s_j))$, while $\mathcal{F}(s_i) \neq \mathcal{F}(s_j)$.

To offset the value function inaccuracies we employ a lookahead control policy, to envision future states. The motivation for using search is derived from the domain of games such as chess [Hsu et al. 1995] and checkers [Schaeffer et al. 1992] where search is the method of choice in overcoming inaccuracies present in the state valuation function. Unlike the games domain, where one would like to search as deep as possible, the FIMS domain has further inaccuracies (described next) prohibiting the use of deep lookahead.

2.4 Inaccurate Domain Models

Individual computer vision operators can take hours to compute on large images. Combined with the exponential number of operator applications needed

for the lookahead long running times make the actual operators unusable for envisionment. Thus, approximate versions of such operators comprising the domain model $\widetilde{\delta}$ are employed within the lookahead. Consequently, such simplified models are inaccurate and, therefore, unable to foresee future states precisely. Sequential noisy predictions introduce compounded errors into envisioned states and thus into $\widetilde{V}^*$ values which offset the benefits of deep lookahead searches.

2.5 Solution Strategy

FIMS clearly exemplifies the typical problems commonly encountered in real world domains. The need for state abstraction to reduce the state space, the existence of value functions inaccuracies, and errors in the state transition function prevent the applicability of classical MDP and search methodologies. However, a fine tuned combination of feature functions, state evaluation functions, domain model approximations combined with dynamic lookahead depth selection may in fact produce near optimal results. Naturally these problems motivate an investigation into the conditions under which the use of a lookahead control policy is most applicable.

3 Experimental Domain

Many Reinforcement Learning projects have used the grid world domain as a test bed [Sutton, Barto 2000]. In the following we refine the classical definition by introducing state abstraction in a particular fashion compatible with the real-world FIMS domain. This compatibility enables further scalability studies via a transition from the grid world to the actual FIMS system. The refined grid world is referred to as the Maze Domain. The *maze* is represented by an $N \times N$ two-dimensional matrix M with each cell being in two possible states: *empty* $M(x, y) = 0$ or *wall* $M(x, y) = 1$. There is a single *agent* in the maze that can occupy any of the empty cells. Additionally, one of the empty cells (x_g, y_g) contains a *goal*. The maze is surrounded by a solid wall preventing the agent from wandering off the map. An *agent's raw state* is a pair of coordinates (x, y) with $0 \leqslant x, y \leqslant N - 1$. Formally: $S = \{0 \ldots N - 1\} \times \{0 \ldots N - 1\}$.

The set A of agent's actions is comprised of λ equally spaced directions and a special action 'quit'. Each of the λ move actions transports the agent along a ray shot from the agent's current location at the angle of $\frac{360 \cdot a}{\lambda}$ degrees where $0 \leq a < \lambda$. The Euclidean distance travelled is upper-bounded by τ and walls encountered (Figure 2). We represent this with a deterministic state transition function $\delta_d(s, a) = s'$.

The MDP *immediate reward* function is defined as:

$$r(s, a, s') = \begin{cases} \mathcal{R}(s'), & \text{if } a = \text{'quit'}, \\ -\|s, s'\|, & \text{otherwise.} \end{cases} \tag{1}$$

Here the cost $\|s, s'\| = \sqrt{(x_s - x_{s'})^2 + (y_s - y_{s'})^2}$ of action a is defined as the Euclidian distance between the new and the old cells. The terminal reward $\mathcal{R}(s)$

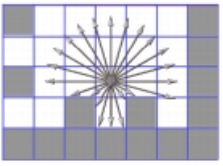

Fig. 2. Agent's actions in the maze domain ($\lambda = 24, \tau = 2$). Cells occupied by walls are inaccessible (shown in grey).

of quitting in state s is defined as: $\mathcal{R}(s) = \Theta - E_{sp}(s, s_g)$,where Θ is a constant, and $E_{sp}(s, s_g)$ is the sum of Euclidean distances of steps in the shortest path from state s to s_g. A game terminates whenever the agent executes the 'quit' action or the move quota is exhausted.

3.1 State Abstraction

Within the maze domain, the state abstraction is simulated via *fixed tiling*. Specifically, if the raw state is $s = (x, y)$ the abstracted state is given as:

$$\mathcal{F}_k(x, y) = \left(\left\lfloor \frac{x}{k} \right\rfloor \cdot k, \left\lfloor \frac{y}{k} \right\rfloor \cdot k \right) \tag{2}$$

where the floor function $\lfloor\ \rfloor$ returns the integer part of its argument. Parameter $k = 1, 2, \ldots, N$ is the *degree of abstraction*. Effectively, the entire maze is divided into non-overlapping rectangular $k \times k$ tiles.

The underlying motivation of this particular abstraction scheme is compatibility with state abstraction in FIMS. Namely, in a computer vision system like FIMS and ADORE, most operators add an information datum of a different category to the problem-solving state (e.g., add extracted edges to an image). The state abstraction features used for various types of data vary considerably (e.g., average brightness for images and mean curvature for extracted edges) making it unlikely that an operator application leaves the agent in the same abstraction tile. On the other hand, several similar operators can move the agent to a single tile (different from the current one). Furthermore, some computer vision operators are not applicable in certain problem-solving states which is modelled by certain angles being blocked off by the maze walls.

4 Lookahead Control Policy

If state aggregation has the property that a longer sequence of actions is more likely to lead outside an abstracted state, then deeper lookahead plays an important beneficial role. By considering only one action or a short sequence of actions the agent can observe that all reachable states have the same values. Thus in such a case deeper lookahead increases the probability of the agent "seeing" outside the current state abstraction tile and seems to be quite beneficial. Therefore, lookahead can provide the agent with a more accurate guidance

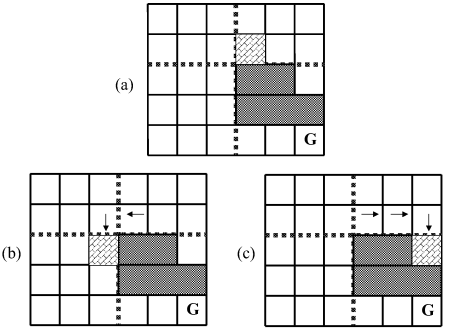

Fig. 3. (a) A maze domain partitioned into four state abstraction tiles (dashed lines). The agent's starting position is the mesh filled cell and the goal position is marked with a "G". Walls are shown as grayed out, solid cells. For this example there are 4 actions (north, east, south, west) available to the agent (i.e. $\lambda = 4$). **(b)** Final position of the agent using lookahead with depth of $p = 1$. **(c)**Final position of the agent using lookahead with a depth of $p = 3$.

function. Is deeper lookahead *always* beneficial? Figure 3, illustrates an example of adverse effects of a deeper lookahead in a simple maze domain. There are 5 deterministic actions: up, down, left, right, and 'quit' available to the agent. The agent has perfect knowledge of its position within the maze but the value function is the same for all states in each abstracted state (or tile). Assume that the agent starts in the mesh filled square of Figure 3a. A *greedy* policy considers only a lookahead of 1 ply deep. Choosing an action under such policy leads the agent to the state denoted by the mesh filled square in Figure 3b (which is a local maximum provided depth 1 lookahead). But if the agent considers sequences of 3 actions (a lookahead of 3 plies deep) it ends up in the state denoted by a mesh filled square of Figure 3c. This cell is a much worse state than the terminal state in Figure3b. Thus deeper lookahead results in a policy leading the agent away from the goal rather than bringing it closer. This example illustrates adverse impact of abstracting together states with very different V^* values. In general, however, we found that deeper lookahead does help. The results consistently demonstrated that deeper lookahead in the face of state abstraction is beneficial with the aforementioned exceptions being rare. Figure 4 demonstrates the agents average performance within a larger (48×48) maze domain with $\lambda = 8$ and a distanse of each action $\tau = 2\sqrt{2}$. Clearly lookahead plays a beneficial role in this case by helping the agent 'see' beyond the abstraction tile it currently occupies.

In contrast to the above results, we found cases where lookahead did neither improve nor degrade the agents performance within the maze domain. Figure 5 shows the performance results of an agent using a machine learned $\widetilde{V}^*$ function. Unlike errors due to state abstraction, errors induced by machine learning do not appear to be diminished by deeper lookahead search.

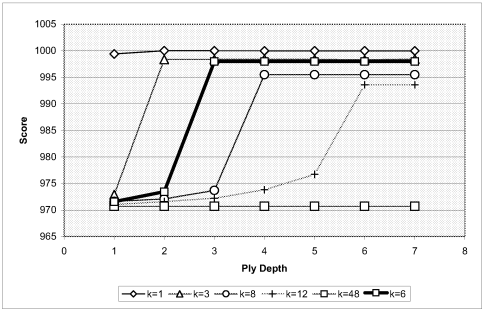

Fig. 4. Reward (score) vs. Ply Depth for experiments with various state abstraction tile sizes (k). Clearly lookahead improves the agent's performance, with optimal ply depth (resulting in maximal reward) proportionally rising with the increase in state abstraction, k.

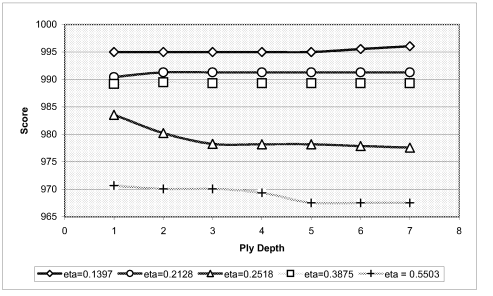

Fig. 5. Reward (score) vs. Lookahead Ply Depth for experiments using machine learned $\widetilde{V}^*$ function. η is the measure of error present within the $\widetilde{V}^*$ with respect to V^*. It is clear there are cases of $\widetilde{V}^*$ where lookahead does neither help or hinder the performance of an agent.

5 Summary and Future Work

This preliminary study focused on identifying the benefits of lookahead control policies within MDP/POMDP environments. Clearly, even for the trivial cases presented, lookahead can have significantly different effects on the agent's performance with respect to the rewards gained. Such contrasting results motivate further studies to establish concrete rules for when the benefits of lookahead outweigh the computational costs of doing so. Furthermore as seen in experiments presented even without the erroneous state transition (domain) model the benefits of lookahead drop off at specific ply depths depending on the types of inaccuracies present. It is therefore beneficial to establish a concrete testing procedure for determining optimal ply depth within the lookahead control policies.

Acknowledgements. We would like to thank Natural Sciences and Engineering Research Council (NSERC) for their sponsorship of this project.

References

[Bulitko et al. 2000] Bulitko, V., Caelli, T., McNabb, D. 2000. Forestry Information Management System (FIMS): An Introduction. Technical Report. University of Alberta.

[Draper et al. 2000] Draper, B., Bins, J., Baek, K. 2000. ADORE: Adaptive Object Recognition, *Videre*, 1(4):86–99.

[Good 1971] Good, I.J. 1971. Twenty-seven Principles of Rationality. In *Foundations of Statistical Inference*, Godambe V.P., Sprott, D.A. (editors), Holt, Rinehart and Winston, Toronton.

[Gougeon 1993] Gougeon, F.A. 1993. Individual Tree Identification from High Resolution MEIS Images, *In Proceedings of the International Forum on Airborne Multispectral Scanning for Forestry and Mapping*, Leckie, D.G., and Gillis, M.D. (editors), pp. 117-128.

[Hsu et al. 1995] Hsu, F.H., Campbell, M.S., Hoane, A.J.J. 1995. Deep Blue System Overview. *Proceedings of the 9th ACM Int. Conf. on Supercomputing*, pp. 240-244.

[Korf 1990] Korf, R.E. 1990. Real-time heuristic search. *Artificial Intelligence*, Vol. 42, No. 2-3, pp. 189-211.

[Larsen, Rudemo 1997] Larsen, M. and Rudemo, M. 1997. Using ray-traced templates to find individual trees in aerial photos. *In Proceedings of the 10th Scandinavian Conference on Image Analysis*, volume 2, pages 1007-1014.

[Newborn 1997] Newborn, M. 1997. *Kasparov vs. Deep Blue: Computer Chess Comes Out of Age*. Springer-Verlag.

[Pollock 1994] Pollock, R.J. 1994. A Model-based Approach to Automatically Locating Tree Crowns in High Spatial Resolution Images. *Image and Signal Processing for Remote Sensing*. Jacky Desachy, editor.

[Schaeffer et al. 1992] Schaeffer, J., Culberson, J., Treloar, N., Knight, B., Lu, P., Szafron, D. 1992. A World Championship Caliber Checkers Program. *Artificial Intelligence*, Volume 53, Number 2-3, pages 273-290.

[Sutton, Barto 2000] Sutton, R.S., Barto, A.G., 2000. *Reinforcement Learning: An Introduction*. MIT Press.