Projective Visualization: Acting from Experience

Marc Goodman*

Cognitive Systems, Inc.
234 Church Street
New Haven, CT 06510

Computer Science Department
Brandeis University
Waltham, MA 02254

Abstract

This paper describes *Projective Visualization*, which uses previous observation of a process or activity to project the results of an agent's actions into the future. Actions which seem likely to succeed are selected and applied. Actions which seem likely to fail are rejected, and other actions can be generated and evaluated. This paper presents a description of the architecture for Projective Visualization, preliminary results on learning to act from observations of a reactive system, and a comparison of two types of *Case Projection* (how situations are projected into the future).

Introduction

An agent must balance a variety of competing goals. An action which satisfies one goal may cause other goals to become unsatisfiable. For example, while standing on a street corner I may have two competing goals: 1) cross the street, and 2) avoid getting hit by a car. If I step out into the street towards the other side, an oncoming car might hit me. On the other hand, if I stand on the corner, I'm in little risk of getting hit, but I won't be getting to the other side of the street.

One method of selecting an action is to *project* the situation into the future. If I project to a state where both goals are satisfied (I've crossed the street without getting hit), then I know the action is appropriate. On the other hand, if one or more goals are defeated (I get hit by a car, or I fail to cross the street within a certain time period), then I know the action is unlikely to succeed. [Goodman, 1989] uses a simple version of projection where a battlefield commander can evaluate the effectiveness of battle plans by projecting the outcome of the battle. The focus in the battle planning example, as in this paper, is on how observation of previous experience can be used to project and to guide action.

*Thanks to David Waltz and Richard Alterman for useful discussion. This work was supported in part by DARPA under contract no. DAAH01-92-C-R376.

Two approaches to representing experience are: 1) to track perceptual observations of an environment (a *concrete* approach), and 2) to extract semantically or causally relevant features from the experience and represent those (an *abstract* approach). For example, Troop movements could be represented either as a series of observations of the positions and orientations of individual soldiers (the concrete approach) or as summary information about the number of soldiers, the type of maneuver (i.e. frontal, enveloping, etc.), and their overall distance to the front (an abstract approach). Previous CBR work has focussed on applying abstract representations of experience to planning [Hammond, 1986; Alterman, 1988; Alterman *et al.*, 1991].

A concrete approach to representation is preferable because of:

- **Knowledge Engineering Difficulty.** An abstract approach requires that an expert interpret each situation to extract the relevant semantic and causal features. A concrete approach stores experience directly from sensor readings or a perceptual system without human intervention. As the number of cases in a system rises, the bottleneck created by an abstract representation becomes severe.
- **Information Loss.** An abstract representation discards features which are deemed irrelevant by a knowledge engineer or domain expert. Unfortunately, what the knowledge engineer or domain expert decide is irrelevant may turn out to be quite relevant. A concrete representation tracks all available information lessening the possibility of information loss.
- **Psychological Evidence.** [Gentner, 1989], suggests that the most common type of reminding is based on surface-level features rather than abstract features. A representation which facilitates surface-level remindings at the expense of additional required work for abstract remindings is, therefore, plausible. A concrete representation satisfies these requirements since surface level features are immediately available for retrieval, but abstract features must be extracted from the representation for ab-

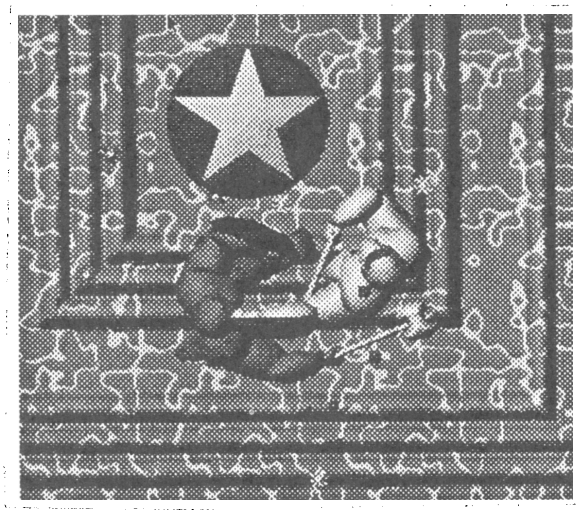

Figure 1: A picture from *The Bilestoad.* Should the agent (light grey) continue fighting or disengage from its opponent?

stract reminding. [Waltz, 1989] points out that since the perception-cognition-action loop typically takes 100ms or less, not much time is available for extracting abstract features from the perceptual world. A concrete representation avoids this problem by dealing directly with perceptual features.

The method of projection presented in this paper uses a concrete representation of a process or activity to create concrete projections into the future. Since these projections have the same structure and content as perceptual observations of the world, we consider this a form of imagery and call the technique *Projective Visualization* by analogy to a form of spatial, visual imagery in humans.

This paper will present results which indicate that Projective Visualization can be used to learn to act through observation. This paper also presents some experiments on evaluating different techniques for *Case Projection*, the underlying engine through which imagery is achieved.

Test Domain

A new version of the video game entitled *The Bilestoad*, which was published by Datamost Software in 1983, serves as the test-bed for the projective visualizer (a picture of a game in progress appears in Figure 1). The game is a simulation of combat between two gladiators armed with battle axes and shields on islands which contain strategic devices. A reactive system was designed and built as an opponent for human players, and serves as a standard of performance for the projective system.

At each frame of the game, approximately 200 pieces of perceptual information about the agent and its environment are collected and processed. This information includes the Cartesian coordinates of each joint in the bodies of the agents, absolute angles of those joints, state information about the joints and the agent overall, the location of devices in the simulation, motor-control of the agents, etc.

The agent must satisfy conflicting goals to succeed. Consider a scenario where the agent wishes to engage a fleeing opponent. One course of action is to chase the opponent. Another course of action is to navigate to a transportation device, and then chase the opponent. Projection can be used in this situation to determine that approaching the opponent directly will fail to cause engagement, since the opponent is moving away from the agent at the same rate of speed as the agent is approaching. The agent visualizes itself chasing the opponent until the opponent reaches its goal. Navigating to the transportation device will result in engagement, and the agent visualizes itself catching its opponent. Action must be initiated quickly, since the longer the agent spends deciding what to do, the more of a lead the opponent will have.

The agent must also successfully anticipate the actions and objectives of its opponent. Consider a situation where the agent is pursuing its opponent and the opponent begins to alter its bearing. If the agent reacts to this by approaching its opponent directly, the agent may reduce the distance to its opponent but allow the opponent to circle around the agent. On the other hand, if the agent projects the effects of the opponent's bearing change, visualizes the opponent circling around, and maintains a position between the opponent and the goal device, the agent will prevent its opponent's escape. Hence, reacting to the opponent's actions is not enough, the agent must anticipate as well.

Case Projection

In the Battle Planner, abstract representations of historical battles were used to project the outcome of a new battle. The system induced a discrimination tree where features of the historical cases (such as ratio of attacking and defending troops, air superiority, amount of artillery, etc.) which were good predictors of the winner of the battle served as indices for case retrieval. There are several techniques for inducing such a discrimination tree, including ID3 [Quinlan, 1986], CART [Brieman *et al.*, 1984], and Automatic Interaction Detection [Hartigan, 1975]. Though each algorithm has slightly different characteristics with respect to convergence, noise sensitivity, and sensitivity to representation, they all share the characteristic of asymptotically approaching an accuracy limit as the number of examples increases, and they can be automatically reapplied as more experience is gathered without additional knowledge engineering.

Projecting a single result (e.g. the outcome of a battle) from a situation description in one step is inappropriate for the following reasons:

- **Sensitivity to Initial Conditions.** Consider Ben

Franklin's old adage, "For want of a nail, the shoe was lost; For want of the shoe, the horse was lost..." which demonstrates that minor differences between situations can compound as the situation evolves. Capturing this sensitivity to initial conditions in a single discrimination tree requires a fairly exhaustive set of examples. The world, being a complex place, makes having an exhaustive set of examples impractical.

- **Interactions between Features.** In battle planning, having more tanks than your opponent is generally beneficial. However, if you are fighting in marshy, heavily wooded, or urban terrains, your tanks will get bogged down, and having more tanks may actually hurt you. To learn this interaction with a single discrimination tree requires that you see cases with many tanks in urban terrain, few tanks in urban terrain, many tanks in flat terrain, few tanks in flat terrain, etc. Once again, the system requires a fairly exhaustive set of examples.

A Projective Visualizer avoids these problems by creating a projected situation which is temporally (and causally) near to the current situation. Instead of simply jumping to the outcome of the battle from the initial conditions, the system simulates how the situation evolves over time. Each step of projection lessens the causal distance between a situation and its conclusion. In The Bilestoad, the agent might have 2 hits of damage to the shoulder of its arm which holds the axe, while its opponent only has 1 hit of damage. Even if we know the relative locations and orientations of the agent and the opponent it may be hard to say which will win the fight. It's easier to say that since the opponent's axe is in contact with the agent's shoulder, on the next time step the agent's shoulder will have 3 hits of damage. Meanwhile, the opponent continues to have only 1 hit of damage, since the agent's axe is not in contact with its shoulder. Continuing to project, we visualize the agent with 4 hits, 5 hits, 6 hits, 7 hits, until the agent's axe arm is severed at the shoulder. At this point, it's much easier to say that the agent will lose its battle. We have reduced the causal distance between a situation and its conclusion by projecting the situation forward in time.

Projective Visualization uses *Case Projection* as an underlying engine for imagery. Case Projection is the process of creating a projected situation from a current situation. A Case Projector is built from a library of experiences by inducing a separate discrimination tree for each feature of a situation. The decisions in the tree correspond to features of the current situation which are good predictors of the *next* value of the feature we wish to predict. Given k concrete features which represent a situation, we induce k discrimination trees.

Projection consists of traversing these k discrimination trees with a case representing the current situation, making a prediction on the next value of each feature, and storing these predictions into a new case. This projected case then serves as a basis for further projection. Hence, a current situation can be driven forward arbitrarily far, at a cost of compounding errors from earlier retrieval. This process continues until the system is able to make an evaluation of the projected situation (is the projected result good or bad), or until some pre-defined limit on projection is reached.

Cases include observations of the world as well as "operators." Each case in the system represents a fine-grained approximation of a continuously evolving situation, in the same way as a motion picture is a fine-grained approximation of the recorded experience. In any particular case, agents are in the process of carrying out actions. For example, in a situation representing a quarterback throwing a football, in one case the quarterback might be moving his left leg backwards, moving his right hand forward, and squeezing with the fingers on his right hand. "Operator" refers to this pattern of control signals. There is, therefore, a one-to-one mapping between operators and cases. Note that a linguistic term like "throw" or "dodge" actually corresponds to a sequence of cases and their corresponding operators.

The indices used for Projective Visualization in The Bilestoad and in comparisons of techniques for Case Projection are generated inductively from the case library with an extended version of the Automatic Interaction Detection algorithm [Hartigan, 1975]. Induction is guided and enhanced in a variety of ways, including methods for enriching the case representation and methods for preselecting good discriminators. Each of these enhancements reduces the amount of experience needed to reach a given level of accuracy, but does not change the property that accuracy will asymptotically approach a fixed limit. Therefore, the same ultimate effects as reported in this work can be reproduced using off-the-shelf versions of CART, ID3, AID, or other learning algorithms, even without these enhancements, but a larger base of examples may be needed.

Dynamics of Projective Visualization

Projective Visualization layers on top of Case Projection to provide a framework for controlling action. The basic idea of Projective Visualization is to pick a likely operator to perform and run the situation forward into the future until one of three things happens: 1) the system will run into an obviously bad situation, in which case the operator should be avoided and another operator tried, 2) the system will run into an obviously good situation, in which case the operator should be applied and real action taken in the world, 3) the system projects the case farther and farther into the future, without conclusive evidence one way or another. If the system is unable to reach a conclusion by the time it's projected a prespecified amount of time, it can make a guess as to whether the operator is good or bad by retrieving the closest case and chasing pointers until the case was resolved either positively or negatively.

Case Retrieval can be used to select likely operators. Given all the cases which lead to the successful satisfaction of a goal, we build a discrimination tree where the discriminations in the tree are features of the current situation which are good indicators of the type of operation. Suggesting an operation to apply becomes a matter of traversing this discrimination tree and collecting operations that were applied in the retrieved cases (we refer to this process as *Action Generation*). Different discrimination trees built from cases where different goals were satisfied can be used to suggest different operators. For example, in The Bilestoad, one action generator may consist of all the actions which led directly to using a transportation device, another action generator may consist of actions which led to killing the opponent, a third action generator may consist of actions which prevented the agent from taking damage, etc.

Evaluating whether a situation is good or bad can be treated as a case retrieval task. For example, in battle planning, a projected situation where 90% of your soldiers have been killed might retrieve a different battle where 90% of the soldiers were killed, and the mission failed. Since the mission failed in this retrieved case, we conclude that the mission will fail in the current (projected) situation as well, a bad thing. In a process control domain, where continuous feed roasters are being used to roast coffee beans, evaluation might be based on Neuhaus color readings on the ground beans as well as by moisture content of the beans. If the projected coffee beans retrieved cases where the color and moisture content differed from ideals specified in the roasting recipe, then the evaluation would be negative. In The Bilestoad, retrieving cases where the opponent is dead indicates a positive outcome, and retrieving cases where the agent is dead indicates a negative outcome.

The "branching factor," or number of different paths the system pursues in projection can depend on both the number of different operators which are suggested by a set of retrieved cases and the number of possible projected values for key features of the case. [Goodman, 1989] indicates that by ignoring predictions where only a few examples are retrieved yields a higher overall accuracy. This suggests that a measurement of confidence can be generated from the number and distribution of retrieved cases, and this confidence measure can be used to prune the search tree so that only the most likely paths are explored.

When the system is in a time-critical situation, response-time can be improved by reducing the window of projection (how far the situation is projected into the future) as well as by considering fewer alternative actions (reducing the branching factor). Such a system behaves more and more as a purely situated or reactive system would [Suchman, 1987; Agre and Chapman, 1987; Chapman, 1990]. On the other hand, when more time-resources are available, the system can project the effects of its actions farther and farther, and consider a greater number of alternatives, causing the system to exhibit more and more planful behavior.

This type of system can learn in several ways. First, through observation of an agent performing a task, it can learn to suggest new candidate actions. It can also refine its ability to suggest actions based on improvements in indexing as more experience is gathered. Next, by carrying along projections and matching them against the world as the situation plays out, it can detect where projection has broken down. Projection can then be improved by storing these experiences into the appropriate projectors and generating new indices to explain these failures. Finally, it can improve its ability to evaluate situations by noting whether its evaluations match outcomes in the real world. Through observation, it can both add new evaluations as well as refine its ability to evaluate by improving evaluation indices.

Evaluation of Projective Visualization

Approximately 28,000 frames of two reactive agents competing in hand-to-hand combat were captured as raw data from The Bilestoad. This represents approximately 1 hour and 18 minutes of continuous play. A case projector was built using this data which predicts whether the agent will cause damage to its opponent in the next frame. Cases representing situations wher the agent caused damage to its opponent in the nex frame were selected (approximately 3,300 such case existed in the 28,000 frames), and used to build an ac tion generator. The action generator consists of eigh sets of indices, one for each control signal for the agent These control signals include moving the axe clockwise moving the axe counterclockwise, moving the shield clockwise or counterclockwise, turning the torso clockwise or counterclockwise, and walking forward or backward. Given these controls, there are $3^4 = 81$ significant patterns of control signals which the agent can receive. Cases representing situations where the agent successfully avoided damage (as indicated by a frame where damage was taken followed by a frame where no damage was taken) were selected (amounting to 1,100 cases out of the original 28,000 frames) and used to build an additional action generator. Projection was used to mediate between these two of action generators, such that if the agent could cause damage to its opponent in the next frame it would, otherwise it would use the avoidance action generator.

The mean difference in score between the projective agent and the reactive agent favored the projective agent by 18.01 points, with a standard error of 12.93 for 1000 games. Since the mean difference of 18.01 is within the 95% confidence interval of 25.34, we accept the null hypothesis that there is no significant difference between the reactive and projective agents. We have, therefore, successfully learned to act as well as the reactive agent through observation of the reactive agent, a result which Chapman was unable to achieve

in [Chapman, 1990].

Evaluation of Case Projection

If we wish to project a situation k steps into the future (which we refer to as a *Projection Window* of k steps), two contrasting techniques exist for performing this projection. The first technique, called *Projective Simulation*, projects the situation forward 1 step, then projects the projection 1 step, and so on, until the situation has been projected k steps. The second technique, termed *One-Step Projection*, performs one retrieval for each feature of the situation, to predict what the value of that feature will be k steps in the future. For example, if we wish to project a situation 5 steps into the future we can perform 5 steps of projection of 1 time unit each (Projective Simulation), or perform 1 step of projection of 5 time units (One-Step Projection).

One of the exhibits at the Boston Museum of Science is a big enclosed table with a circular hyperbolic slope in it, leading to a hole. Steel balls are released from the edge of the table and go around and around the slope until their orbits decay enough that they fall into the hole. One of the things that makes this exhibit fun to watch is that when the ball is heading toward the hole it accelerates, and if it misses the hole it "sling-shots" around, leading to much visual surprise and merriment. After watching, however, one quickly learns to predict how the steel ball will move around the track, a form of case projection. One-Step Projection and Projective Simulation were compared in the domain of prediction of the position of an object in Newtonian orbit around a gravity source (which is what the exhibit represents). For a full treatment of the physics involved, see [Halliday and Resnick, 1978].

For the following tests, the Gravitational Constant, the mass of the gravity source, the mass of the object in orbit and the initial velocity of the object were chosen to yield a reasonable level of granularity in the orbit. The initial position of the object was varied randomly within a fixed range. For the values chosen, the average number of time steps for an object to complete an orbit around the gravity source was 44.75. A training set of 20 orbits was created by randomly selecting the initial position and deriving subsequent positions from a quantitative model, yielding 895 cases. The set of features on each case was limited to the current position of the object in two-dimensional Cartesian coordinates, the change in position from the previous observation for each coordinate, and the previous change in position for each coordinate. No reference to the position of the gravity source, the masses of the object and source, or the laws of physics, were given to the system for use in projection.

Figure 2 shows the percentage of orbits where One-Step Projection was more accurate than Projective Simulation, for Projection Windows of varying size, given a training set of 20 orbits. When the Projection Window is equal to 1 time step, Projective Simulation and One-Step Projection are functionally equivalent. For Projection Windows between 2 and 20, 300 initial starting positions were randomly selected and the mean error per orbit was determined and compared. The resulting percentages are accurate to within 4.8%, given a 95% confidence interval based on a one-tailed distribution. For Projection Windows between 21 and 33, 100 initial starting positions were used, yielding a 95% confidence interval of at most 8.3%, based on a one-tailed distribution.

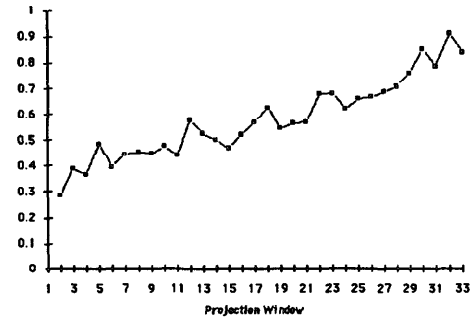

Figure 2: Percentage of Trials where One-Step Projection is More Accurate than Projective Simulation, Size of Training Set=20 Orbits.

Projective Simulation was more accurate than One-Step Projection for small Projection Windows. Specifically, for Projection Windows between 2 and 11, the null hypothesis that the two methods are equivalent could be rejected for 8 of the 10 points with 95% confidence. On the other hand, One-Step Projection was more accurate than Projective Simulation for large Projection Windows. For Projection Windows between 22 and 33, the null hypothesis could be rejected for all points with 95% confidence.

Figure 3 shows the mean of the mean error per orbit for Projective Simulation and One-Step Projection with training sets of size 5 Orbits and 20 Orbits, as the size of the Projection Window is varied. These data points are based on a random sample of 200 initial positions. The error rate is roughly linear in the size of the Projection Window for small Projection Windows. The slope of these linear components of the errors is slightly smaller for Projective Simulation, hence Projective Simulation is more accurate for small Projection Windows. For larger Projection Windows, the change in error rate for One-Step Projection begins to decrease, and One-Step Projection becomes more accurate than Projective Simulation.

Discussion

The central difference between One-Step Projection and Projective Simulation explains these results. At

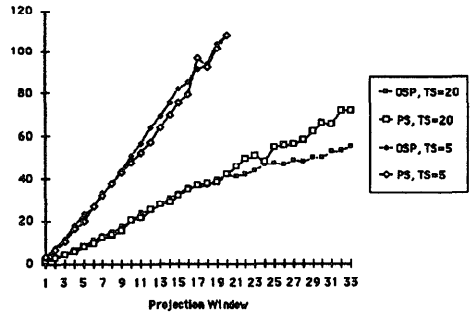

Figure 3: Error VS. Projection Window for Training Sets of Size Five Orbits (TS=5) and Twenty Orbits (TS=20) for One-Step Projection (OSP) and Projective Simulation (PS).

each step in Projective Simulation, the system is free to select new previous situations for subsequent projection, where One-Step Projection is forced to follow one set of previous situations to their conclusion. This benefits Projective Simulation in the short term, since Projective Simulation can account for cascading differences that would render any one previous situation obsolete. In other words, as minor initial differences between the current situation and a retrieved situation begin to compound, Projective Simulation automatically chooses better predictors of future values. One-Step Projection does not have this flexibility, and must follow one set of precedents no matter how much the current situation begins to deviate. Hence, the slope of the error rate for Projective Simulation will be less than the slope of the error rate for One-Step Projection.

On the other hand, One-Step Projection offers a benefit which is lacking in Projective Simulation. One-Step Projection guarantees that the projected situation will be *internally consistent.* Specifically, the space of possible projections is bounded by the case base, and since One-Step Projection relies on a single set of precedents, the projection can never "break out" of these bounds. For domains where there are inherent limitations on values of features, this imposes a maximum error rate for any particular orbit. For example, in the orbit domain, given a range of initial positions and a fixed initial velocity, the minimum and maximum Cartesian coordinates of an object are bounded in a fixed space. As the size of the Projection Window grows, the mean error rate will approach the error resulting from a random selection of points from the fixed space for projection, which is constant for sufficiently large samples. Hence, the error rate for One-Step Projection will asymptotically approach a constant limit. The constant will, of course, depend on the particular domain. The flexibility inherent in Projective Simulation allows it to break out of these bounds, and the error rate retains its linear characteristic.

Conclusions

We have demonstrated that systems can learn to project the effects of their actions through observation of processes and activities. We have demonstrated two techniques for projection, One-Step Projection and Projective Simulation, and have indicated that Projective Simulation is appropriate for projecting effects in the short-term and that One-Step Projection is appropriate for projecting effects in the long-term. Finally, we have demonstrated that a system which controls an agent acting in an environment can benefit from projection, even when projection is not 100% accurate.

References

Philip E. Agre and David Chapman. Pengi: An implementation of a theory of activity. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 268–272, 1987.

Richard Alterman, Roland Zito-Wolf, and Tamitha Carpenter. Interaction, comprehension, and instruction usage. *Journal of the Learning Sciences*, 1(4), 1991.

Richard Alterman. Adaptive planning. *Cognitive Science*, 12:393–421, 1988.

L. Brieman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Wadsworth, 1984.

David Chapman. *Vision, Instruciton, and Action.* PhD thesis, Massachusetts Institute of Technology, Cambridge, Mass, April 1990.

Dedre Gentner. Finding the Needle: Accessing and Reasoning from Prior Cases. In *Proceedings the Second DARPA Workshop on Case Based Reasoning*, pages 137–143, 1989.

Marc Goodman. CBR In Battle Planning. In *Proceedings the Second DARPA Workshop on Case Based Reasoning*, pages 312–326, 1989.

David Halliday and Robert Resnick. *Physics, Parts 1 and 2.* John Wiley and Sons, 1978.

Kristian J. Hammond. CHEF: A model of case-based planning. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 267–271, 1986.

J. Hartigan. *Clustering Algorithms.* John Wiley and Sons, 1975.

J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

Lucy A. Suchman. *Plans and Situated Actions.* Cambridge University Press, Cambridge, 1987.

David Waltz. Is Indexing Used for Retrieval? In *Proceedings the Second DARPA Workshop on Case Based Reasoning*, pages 41–44, 1989.