Towards Manifold Learning for Gamebot Behavior Modeling

Christian Thurau
Applied Computer Science
Bielefeld University
33615 Bielefeld, Germany
cthurau@techfak.uni-bielefeld.de

Christian Bauckhage
Centre for Vision Research
York University
Toronto, Ontario, M3J 1P3
bauckhag@cs.yorku.ca

ABSTRACT
Traditionally Computer Game Agent behaviors are generated by top-down approaches like finite state machines or scripts. So far, however, this had only mediocre success in creating life-like impressions. The bottom-up approach of imitation learning for agents has become very popular in recent robotics research and, in earlier work, we already discussed how imitation learning may apply to the programming of life-like computer game characters. However, so far we ignored problems concerning high dimensional state spaces for the most part, although behavior execution and learning takes place in such spaces.

In this paper, we investigate the usage of non-linear dimensionality reduction for gamedata. We therefore focus on the aspect of topological game world representations and their dimensionality reduced counterparts. Dimensionality reduction is achieved by learning manifolds using Locally Linear Embedding. A mapping between data and embedding space is realized by Radial Basis Function interpolators. Experiments focus on movement path calculation and comparison in 3D and 2D embedding space world representations. The results indicate certain problems inherent to this approach but nevertheless justify further investigations.

1. INTRODUCTION
Over the last years, Computer Games have changed their appearance from simple reaction games to complex world simulations. Games are becoming ever more and more realistic. Interestingly that only holds for visuals and simulation of physics. Up to now, neither academic AI research nor the game industry managed to create convincing, human-like artificial players (gamebots). Nevertheless, there is a huge demand for virtual game inhabitants who could create life-like impressions and thereby add what is left to the realism of today's game worlds.

The commonly used underlying techniques in Computer Game AI are finite state machines and preprogrammed scripts which both highly depend on the effort with which they are programmed. The more time is spend on designing state machines and anticipating all possible situations and appropriate reactions, the more convincing the artificial agent will be. Both methods are not well known for their generalization capabilities, therefore they do not seem to be too well suited for the demands of increasingly complex games. If a human player acts different from what is expected by the game programmers, game agents are very likely to show inadequate, if not "dumb" behavior.

This might be different if artificial game players could learn from their human counterpart. Imitation learning, i.e. the process of learning from observing humans, recently attracted a lot of attention in the humanoid robotics community [2]. Up to now imitation learning in robots is mostly targeted at simple reactive behaviors (e.g. imitation of a tennis swing or button pressing), whereas we have to look for more complex behaviors (tactical and strategical) in the Computer Game AI domain.

The special situation in games justifies high expectations when it comes to the imitation of higher level cognitive aspects. Compared to robotics the usage of games as an experimental environment yields certain advantages; Games are virtual; tedious hardware problems and artifacts such as sensor noise that have to be dealt with in robotics do not exist in the Game domain. Despite this apparent simplification of the problem of behavior learning in the Game domain, it is far from being trivial. Human players show a great variety of behaviors when accomplishing game tasks; their behavior is in fact similar to the handling of real world situations. The needed acquisition of training data (observations), usually a demanding task by itself, is granted due to the popularity of on-
line multiplayer games - we can simply reuse recorded multiplayer matches. These recorded matches, so called demo files, can be downloaded for free and in very large numbers from the Internet (besides, we could create demo files by directly recording human players on Internet game servers). The contents of a demo file is the submitted and received network traffic of a specific player, thus they are able to fully reconstruct a recorded match from the observed player’s perspective.

In [9] we outlined that behavior in First-Person-Shooter (FPS) games (we used the game QUAKE II® (see Figure 1)) can be separated into reactive, tactical and strategic behaviors. We managed to train game agents, who learned either one of the behavioral layers from observations of human players [1] [10] [9]. We understood behavior as sequential actions of a player onto specific game- or player-state vectors, where such state vectors typically included the player’s position, important internal variables (health or armor values), and relative positions of external entities (enemy players or dropped items).

Assuming the state of a player \( p \) at time \( t \) to be given by a vector \( \vec{s}_p^t \), an approximation of the player’s next state could be modeled as:

\[
\vec{s}_{p,i}^{t+1} = \vec{s}_p^t + a_i^t(\vec{s}_p^t) + e_i
\]

where \( a_i^t(\vec{s}_p^t) \) denotes the player’s action dependent on the current state and \( e_i \) denotes environmental influences. Of special interest is the function \( a_i^t(\vec{s}_p^t) \), which we tried to approximate based on the observations of reactions of a human player onto game-state vectors. However, in the mentioned approaches we established imitation of behaviors depending onto only a few of the possible state vector variables. This was sufficient for the development of first working approaches, besides, integration of all possible state vector variables did not seem to be feasible. Unfortunately the selection of relevant state vector variables by expert knowledge leads to certain problems, we implicitly assume that the human player decided based on our selection of state variables, which is very unlikely to be true.

For better imitation of human behaviors in games, we need to consider a greater variety of game-state variables. But analyzing high dimensional data suffers from the curse of dimensionality, the required number of samples grows exponentially with the dimension of data points. Consequently, finding a more optimal, dimensionality reduced representation of game-state vectors is necessary.

We can not expect linear dimensionality reduction methods, such as Principal Component Analysis (PCA) to discover usable subspaces. E.g. reducing 3D world coordinates (which are needed at every time in a 3D computer game) to 2D using PCA can only be successful, if the game already takes place in a 2D subspace, otherwise the loss in information would be too great to allow for appropriate reconstruction of coordinates.

Recently nonlinear dimensionality reduction methods, such as Locally Linear Embedding (LLE)[7] or Isomap [8], are gaining more and more interest. In this paper we want to exploit the usage of nonlinear dimensionality reduction for game data, to find the intrinsic representation of game-state vectors. For that task we use Locally Linear Embedding and apply it to a player’s position based representation of the gameworld, effectively reducing it to a 2D representation. To make use of the dimensionality reduced coordinates, we compute topological representations (waypointsmaps). Thereby we show, that moving on a 2D waypointmap in the embedded space can be used for real movements in 3D.

However, future work will focus on learning manifolds of higher dimensional game-state vectors, which would be what we are really looking for.

2. MANIFOLDS AND GAMES

Learning manifolds aims at finding intrinsic representations of high dimensional data, and thereby simplifying data representation by mapping them to a lower dimensional space. What was so far applied to recognition of facial expressions [7] or gait classification [3] could maybe also applied to game data. If so, we could make the task of learning complex state space dependencies less demanding, and thereby improve situation dependent behavior learning approaches. Jenkins [4] successfully applied Isomap for deriving action and behavior primitives, which is a first hint for the usability of nonlinear dimensionality reduction in the context of behavior learning.

However, so far no one looked for manifolds in games. The positions a player can hold in a 3D game lie on a manifold in the 3D space of all possible positions, that seems obvious. This holds as well for other game-state variables, consider for instance angular positions of enemy players relative to the observed player, which lie on a sphere around the player.

We assume that all possible game situations (or game-states) in a high dimensional space effectively lie on a manifold and could be embedded using LLE or Isomap. Thereby the intrinsic representation could be interpreted as a projection of state data onto it’s lower dimensional game situation space, which would certainly simplify state dependent behavior learning approaches.

2.1 Manifold Learning

Locally Linear Embedding is a popular method for finding compact representations of high-dimensional data [7]. The basic idea of LLE is to project high dimensional data onto a neighborhood preserving lower dimensional space. The projection is done in two steps. First by finding reconstruction weights for each data point and a number of neighbors, so that the reconstruction cost function \( \psi(W) \) is minimized:

\[
\psi(W) = \sum_i \left( \sum_j |W_{ij} \vec{X}_i - \vec{X}_j|^2 \right) \]

, where \( W_{ij} \) denotes the linear coefficient for the reconstruction of the high dimensional vectors \( \vec{X}_i \) by it’s neighboring vectors \( \vec{X}_j \). The weights are constrained by setting \( W_{ij} = 0 \) for \( X_i \) not being a neighbor of \( X_j \), and by enforcing \( \sum_i W_{ij} = 1 \) for the linear coefficients of neighbors of their target reconstruction vector. According to [7] the weights that minimize the reconstruction error are, because of the enforced constraints, invariant to rotations, scalings and translations of the data points and neighbors. Therefore the weights can reconstruct data points in a lower dimensional space, by minimizing the embedding cost function \( \psi(Y) \):

\[
\psi(Y) = \sum_i \left( \sum_j |W_{ij} \vec{Y}_i - \vec{Y}_j|^2 \right)
\]

The second cost function is optimized for the lower dimensional embedded vectors \( \vec{Y} \) while using the already known weights \( W_{ij} \).

In [7] it was shown that both cost functions can be effectively minimized, and by doing so high dimensional is projected onto a lower dimensional space.

However, in order to find embedded coordinates for new data samples, we have to establish a mapping from data to embedding space. Besides, we need the reverse mapping as well, to project vectors from embedding to data space. Similar to [3] we use Radial Basis Functions (RBF) interpolator for that task, for further reading on RBF we therefore recommend [3].

1 Good examples for such games are the so called pseudo 3D shooter games, e.g. DOOM I.
3. LEARNING WAYPOINTMAPS

For a representation of the virtual world a grid based representation, a waypoint map, is learned using a Neural Gas [5] cluster algorithm. Neural Gas clustering is basically an improved k-means algorithm showing good performance when it comes to topology learning [6]. The training data used for Neural Gas learning consists of all positions \( \vec{p} = \{ x, y, z \} \) (or \( \vec{e} = \{ x, y \} \) in embedding space) a human player held during various plan executions, thus staying very close to the actual human movement paths by only taking visited places into account.

Applying a Neural Gas algorithm to the player positions results in a set of position prototypes, places a player visited frequently. The number of prototypes depends obviously on the 3D map size. In a second step neighboring prototypes are interconnected. The edges connecting the nodes are not computed using the classical hebbian topology learning [6]. Drawing edges based on concurrent activity would certainly lead to problems on one-way paths. Therefore we only connected two nodes, if there was an observable player movement from one node to the other. This finally results in a topological map representation or waypointmap. The edges are stored in a \( n \times n \) matrix \( C_{ij} \in \mathbb{N}^+ \) where \( n \) denotes the number of units/neurons. If \( C_{ij} = 0 \) unit \( i \) is not connected to unit \( j \), if \( C_{ij} = 1 \) unit \( i \) is connected to unit \( j \). Figure 3 shows two resulting waypointmap, in embedding and 3D space.

Using a nearest neighbor criterion, every position on the 3D map can be assigned to a single node. A movement from one node to another is only possible if a connection between the two exists, and can be easily translated into “real” movements in the 3D map.

4. EXPERIMENTS

We carried out a number of experiments, to see to which degree LLE could be useful for dimensionality reduction in game data. The game we used for that matter was the well known First-Person-Shooter QUAKE II®. which is a widely used scientific testpad for Game-AI and Game-Learning approaches.

Instead of focusing on the whole state space, we concentrated on the positions \( \vec{p} = [p_x, p_y, p_z] \) a player held during a match. Therefore we computed waypointmaps in the reduced and full state space and compared them.

In particular we were interested in the question if waypoints in a 2D embedding space are usable, so that paths can be computed and effectively retransferred to movements in 3D. Waypointmaps were calculated as follows:

1. Learn manifold by application of LLE to the 3D position \( \vec{p} = [p_x, p_y, p_z] \) recordings of a human player. This leads to embedded positions \( \vec{e} = [e_x, e_y] \)

2. Establish mappings from embedding to 3D space and vice versa using RBF interpolator, which is necessary for transformation of 2D embedding space paths to 3D paths.

3. Cluster waypoints using Neural Gas clustering in both 3D and embedding space.

Figure 3 shows two resulting waypointmaps. Comparison of waypointmaps was done by computing paths in both spaces, using Dijkstra’s algorithm for computation of shortest paths. Since pathfinding in 3D is not much more complex than in 2D, we did not look for any general improvements. It is more a first shot at the whole problem of estimating and finding intrinsic representations of higher dimensional spaces in game data, for the later use in behavior learning.

It showed, that 3D movements could be reduced to movement on a 2D manifold, unfortunately not in all cases and not as robust as we hoped. It became clear, that clustering of waypoints in embedding space does not necessarily lead to usable waypoints. Where some paths could be calculated efficiently, others would have ended up in chaotic movements in 3D. A condensation of player positions in embedding space appears to be very sensitive to parameter selection in Neural Gas clustering (number of waypointmap nodes) and LLE (number of neighbours) and RBF (number of centroids) application. Effectively, LLE parameters seem to be the most difficult to estimate, since we could achieve very good mappings and condensation by choosing a sufficient high number of RBF centroids and Neural Gas cluster centers. However, the right number of neighbors for LLE could not be chosen in the same manner and varied among different map sizes and number of data samples used. Where we managed to show overall good results on smaller maps, with less effort on parameter tuning, we had problems estimating the right parameters values for larger game areas to achieve similar good results.

It will be interesting to see, if the assumption of having situation manifolds in game-state space holds for larger or complete state vectors. Nevertheless, the problems in parameter estimation re-
main and are very likely to have a greater impact with increasing state vector dimensionality. But so far this is the best we can do in reducing game-state vectors dimensionalitys and thereby making functional approximation and behavior learning less demanding.

5. CONCLUSION & FUTURE WORK
Dimensionality reduction is a useful mean for making nonlinear function approximation less demanding. Understanding behavior imitation as a problem of function approximation, we proposed the usage of Locally Linear Embedding for reducing game data dimensionality. First successful experiments justify further examinations, but also showed some shortcomings.

In [1] we trained Neural Networks on subspaces of the game state space, thereby creating situation dependent behaviors. One could think about applying the same approach to the simpler embedding state spaces, which could lead to an incorporation of a greater variety of state space variables. In [11] we realized a state dependent execution of movement or action primitives. That lead to smooth, human-like movements in 3D. One of the problems arose from diversity of state space variables, on which primitive execution could depend on. We manually selected variables we considered important, however, we can never be sure if a human player really decided based on our selected variables. Again, projecting states onto its intrinsic dimensionality could make manual selection redundant, and therefore improve results in imitation learning for behavior acquisition.

Acknowledgements
This work was supported by the German Research Foundation (DFG) within the graduate program “Strategies & Optimization of Behavior”.

6. REFERENCES