# Role of Pattern Recognition in Computer Games

Timo Kaukoranta, Jouni Smed
*Department of Information Technology and Turku Centre for Computer Science (TUCS), University of Turku, Lemminkäisenkatu 14 A, FIN-20520 Turku, Finland*
*timo.kaukoranta@cs.utu.fi, jouni.smed@cs.utu.fi*

Harri Hakonen
*Oy L M Ericsson Ab, Telecom R&D, Joukahaisenkatu 1, FIN-20520 Turku, Finland*
*harri.hakonen@lmf.ericsson.se*

## Abstract

*We discuss pattern recognition in the context of computer game. The purpose of pattern recognition is to extract relevant information from the game world. This high level information is needed by a decision-making system, which is responsible for producing actions to the game world. We delineate where pattern recognition can be applied in computer games, what are its roles, and what is expected from it. We discuss how pattern recognition can be utilized in different levels of detail, and how the relation between the synthetic player and the human player affects to the requirements for pattern recognition.*

## 1. Introduction

Computer games (CGs) are an unique application area for pattern recognition. They are the products of their developers, and, at first sight, it is easy to assume that the developers know inherently what phenomena occur in their game world. However, as the worlds in CGs try to resemble more closely the real world, their complexity increases greatly. Consequently, the need for pattern recognition has become more prominent. Also, to implement challenging synthetic opponents computer should recognize the behavior of a human player. In this context, the purpose of pattern recognition is to abstract relevant information from the game world and to construct concepts and to deduce patterns from this information. These concepts are needed for higher level reasoning and decision-making. In CGs, this is associated with artificial intelligence (AI) problems.

There exists a large amount of scientific articles and books on the fundamentals of pattern recognition and its applications to several different problem settings. Often pattern recognition is called classification or identification—or even AI in general. However, pattern recognition is not one particular technique, but it includes several techniques such as string and clustering algorithms. Unfortunately, in the development of AIs for CGs the scope of pattern recognition has not been widely realized. Pattern recognition could provide essential information for the decision-making system, like what phenomena and events have occurred in the observed (dynamic) system or world. Thus, pattern recognition is only a part—but very important one—of whole AI system.

Our motivation is to do pattern recognition ourselves to discern where pattern recognition can be applied in CGs, what are its roles, and what is expected from it. For most parts, we raise questions and provide examples rather than try to propagate certain techniques or approaches; for example, how pattern recognition can be utilized in several levels of detail, or how the relation between the synthetic player and the human player affects to the requirements for pattern recognition. Only after the ground have been laid out, we can start to think more concretely on the terms of algorithms. Unfortunately, this falls beyond the scope of the present paper. Instead, we try to form a coherent picture of the needs of CGs.

The plan of the paper is following. Introduction to the concepts of pattern recognition is given in Section 2. Roles of pattern recognition in CGs are discussed in Section 3. In Section 4, we point out several topics of CGs related to pattern recognition. Finally, conclusions are drawn in Section 5.

## 2. Fundamentals of Pattern Recognition

Pattern recognition techniques are often lumped together with AI techniques. However, pattern recognition could also be seen as a part of whole AI system. Here, we consider an AI system to be composed of two parts: *pattern recognition* and *decision-making*. The task of pattern recognition is to extract relevant information from the data source (i.e., to classify the data into classes that identify

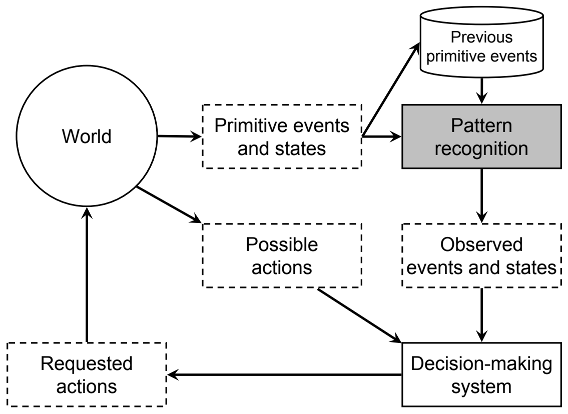

Figure 1: Relations between the world, pattern recognition, and decision-making.

interesting phenomena). In some applications this information extraction is enough but other AI applications may need to perform higher level reasoning. This reasoning can be based on the information extracted by a pattern recognition technique.

The relation of pattern recognition and decision-making in an AI system is illustrated in Figure 1. The world, which can be real or simulated, consists of primitive events and states (i.e., phenomena) that can be perceived. From these and previously stored perceptions, pattern recognition produces higher level observations, which are forwarded to the decision-making system. The world allows a set of possible actions, and the decision-making system has the responsibility to choose an appropriate action based on the information given by pattern recognition.

The simplest form of pattern recognition is pattern matching, which is a process of searching a predetermined pattern from given input. If the pattern is a fixed template, the pattern matching detects the existence of the pattern in the input (e.g., searching a word from text). Pattern can be expressed also as a rule, and in this case the matching is an optimization problem. This optimization bounds the free variables of the rule (e.g., fitting regression line into a data set). Generally, pattern recognition can be used into much more complex problems than pattern matching.

## 2.1. Concepts

Pattern can be defined as a actual phenomenon of the world that we want to recognize. For example, pattern can be an attack or lack of resources. From the data source (i.e., the world) we can perceive simple *measurements*. By assigning some primitive meaning to a measurement we have a *feature*, which can be numerical (e.g., quantity), symbolic (e.g., name or label), or complex (e.g., combination of primitives). Feature selection is a process of choosing the features to be extracted (i.e., relevant data). Extracted features should be discriminative, descriptive, and computationally feasible.

Features can also be called as *symbols*, and a set of symbols forms an *alphabet* $\Sigma$. The selection of symbols depends on the abstraction level of the decision-making system. Moreover, in a hierarchical pattern recognition system the recognized patterns can be symbols for a higher level pattern recognition. To reduce the computational burden of the recognition process, the size of an alphabet can be decreased by quantizing the measurements at the cost of losing information.

A pattern *class* is a set of similar patterns. In other words, the classes should discriminate different types of patterns. The class set can have *a priori* knowledge based on expertise. It can also have *a posteriori* knowledge when it is formed in a learning process. In the classification process an input data is assigned into at least one class based on the extracted features. In fact, recognition can be defined as an ability to classify.

## 2.2. Recognition Methods

Pattern recognition approaches can be categorized into statistical, syntactic (or structural), and neural pattern recognition [8]. Statistical pattern recognition assumes that extracted feature vectors obey some probability density function. Typically, it requires background information from the application area. Syntactic pattern recognition considers interrelationships and interconnections of the features to produce structural information. Neural pattern recognition approaches try to simulate biological neural systems by learning process. They can also be seen as a black-box way to implement statistical and syntactic methods. Indeed, the boundaries between these approaches are vague and fading.

In the design and implementation of pattern recognition system, the task is to absorb significant patterns and rules from the given data. In other words, the task is to *learn the characteristics of the world*. In a supervised learning process, the designer provides correct identification for all inputs, and the process constructs the rules from this information. In an unsupervised learning process, the goal is to define the classes, which is typically performed by a clustering method. This process requires an appropriate training set of features. The variety of actual pattern recognition methods is wide. They utilize, for example, string algorithms, formal languages, clustering techniques, genetic algorithms, neural networks, and fuzzy logic.

## 3. Pattern Recognition in Computer Games

Pattern recognition in CGs can be viewed from several perspectives. In the following we consider first how pattern recognition operates on different levels of decision-making. Then, we describe the roles of pattern recognition in a relation between a synthetic player and a human player. We discuss open game world and story generation, and show how modeling can be used in both prediction and production. Finally, we list some common tasks where pattern recognition could be utilized.

### 3.1. Strategical, Tactical, and Operational Level

By using classical three-level design hierarchy we can divide decision-making problems into three categories: *strategical*, *tactical*, and *operational*. Accordingly, the suitability of pattern recognition methods depends on the level of the decision-making. Let us now categorize common problems and features of pattern recognition.

On the strategical level, pattern recognition works over a long period of time. The amount of data can be large, and, therefore, the main problem is to filter it down to a suitable alphabet. Due to quantization, information is lost and the problem is to ensure that no vital knowledge is left out in this process. The reason for the high volume of data is that it is assembled from all inhabitants, items, and events in the game world. Naturally, we cannot operate with such large set of data. Decision-making in strategical level is speculative (i.e., What-If scenarios). This is usually done offline or in the background. Therefore, it can involve high-quality pattern recognition methods. Another reason to aim at good quality is that the cost of a wrong decision is high on the strategical level.

Tactical level connects the strategical level to the operational level. Because tactical decisions are made more frequently than strategical, pattern recognition has less time to use. Consequently, the quality cannot be as high as on upper level. If the strategy answers what should be done, tactics describes how to actuate it. Action on the tactical level considers a group of entities and their cooperation. Input data for recognition originates from the actors of the operational level, and, therefore, pattern recognition should observe the operational environment.

Operational level is concrete and closely associated with the properties of the game world. Operations are related to atomatory entities, and, therefore, pattern recognition techniques must be reactive, short-term, and real-time. Because the decisions are made for a short term, less accurate pattern recognition does not necessary lead to irrevocable problems. For instance, if the chosen path turns out to lead a dead-end, we can still reroute the path to the destination. Regardless of the level of decision-making, one should always bear in mind Patton's law: A good plan today is better than a perfect plan tomorrow.

### 3.2. Enemy, Ally, or Neutral

Although it may seem more natural to think a computer as an enemy, it may as well be an ally. For example, in a real-time strategy (RTS) game, player's interface could be augmented by implementing a synthetic reconnaissance officer. It could report on enemy movements and even suggest effective counteractions. In this case pattern recognition has to account human perspective, since the result will be given to a human player instead of passing it to a decision-making system.

The third role for the computer player is to act neutral. For instance, in a sports game, like ice hockey or car racing, a commentator could highlight the events and provide background information. Obviously, pattern recognition is needed to recognize what is happening in the game world. Interest in this problem has increased lately; for example, synthetic commentators have been demonstrated in RoboCup [7].

In addtion to synthetic commentators, the computer could act as an autonomous camera director. The camera director programs require sophisticated pattern recognition. For instance, the placement and angle of the cameras in a sports game is usually dictated by the television practice, and the problem is to choose an appropriate camera view (including the angle and zoom) among them all. If there is only a single point of interest, like a human-controlled racing car, the problem is easy; however, if there is action all around the playing field, it is hard to recognize what to show and in what order so that the cuts still form a coherent whole. Things become even more complicated, if the camera can move freely around the game world.

An important neutral role is to act as a referee. However, some rules may be hard to judge because in order to detect their state complex decision-making is required. For example, in soccer the referee can allow the play continue "when the team against which an offence has been committed will benefit from such an advantage" and penalize "the original offence if the anticipated advantage does not ensue at that time" [3]. To implement this in a computer game would require pattern recognition which can interpret (possibly complex) causality between the offence and the subsequent events.

### 3.3. Open Game World and Story Generation

A game is not a story: while a story progresses linearly, a game must provide an illusion of free will [2]. Obviously, the player must have a range of actions to choose from at each stage. More formally, let us consider the

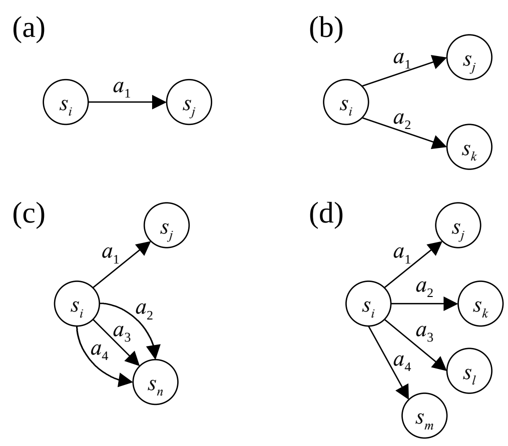

Figure 2: In a game graph, game states are represented as nodes and actions as directed arches. (a) A linear progression (e.g., a story) allows no diversion. (b) Outdegree is the number of arches leaving a node (in this case node $s_i$ has an outdegree of 2). (c) Indegree is the number of arches entering the node (in this case node $s_n$ has an indegree of 3). (d) Although the number of possible actions (i.e., the outdegree of node $s_i$) is the same as in the previous case, each action has now a unique response.

game as a directed graph where the game states are nodes and the possible actions arches (Figure 2). This means that the greater the outdegree (or fan-out) of a node, the more freedom the player has. In this graph, the uniqueness of a response can be measured as the indegree (or fan-in) of a node. For example, assume that the game plot is divided into chapters. Typically, the plot lines of the previous chapter are concluded, and many new plot alternatives are introduced. This means that in the graph the beginning state of the chapter has a large indegree and outdegree. Thus, the game properties can be analyzed through graph concepts (e.g., repetitiveness corresponds to cycles in the graph).

In an *open game world*, the player should have as much freedom as possible and the responses to the actions should be appropriate. An obvious way to implement open game world is simulation. CGs are often likened to simulations, although there are games that are not simulations: We may allow that chess simulates warfare in a highly abstract manner but clearly checkers is not a simulation. Nevertheless, most games—and especially computer games—are simulations, because a resemblance to real-world objects or everyday world (even with a slight touch of fantasy) assists immersion. Indeed, entertainment industry has embraced military simulations because of the realism they provide [1].

The main difference between simulations and games is that games are goal oriented. A flight simulator, for example, is not a game in itself but dogfighting with a flight simulator is. In CGs, a usual approach is to include a story into the game and, as a consequence, limit the simulation. This game-as-a-story approach usually contains a linear—or at most a slightly diverse—plot, where the player has some freedom only between fixed entry points (i.e., the game graph converges to a predetermined state from time to time). Still, most games do not include a story-line nor impose a sequence of events.

Clearly, the game developers would like to include a story into the game. The problem is, as we saw earlier, that the story-generating program must act like a human dungeon master. It must observe the reactions of the crowd as well as the situation in the game, and recognize what pattern fits the current situation: Is the game getting boring and should there be a surprising plot-twist, or has there been too much action and the players would like to have a moment's peace to rest and regroup?

If we consider pattern recognition, the grand challenge for the future game development is story generation in an open game world. Although there have been some efforts towards this direction, they are more like a set of separate missions that have no connection to the dynamic game world itself. Nevertheless, let us propose a design for such a CG using medieval world as an example.

Since we aim at telling a story to the human players, we must ensure that the world around them remains purposeful. We have general plot patterns that we try to recognize in the history and surroundings of a human player. One of these pattern could a parental quest: If the player lives with his or her parents, they can be abducted thus leading the player on a quest to find them. It does not matter, whether the player is an heir to the throne or a humble peasant, the pattern can be found and the story can be set into motion.

We can use lazy evaluation to generate history and non-player characters (NPCs) as they are needed. If during the quest the player chances to meet a robber and manages to befriend with him, we have to line out the history and the character of the NPC; otherwise, he can simply yell "Stand and deliver!", take the player's money, and vanish into the night. In the latter case, the story generator could now recognize the pattern that the player is penniless, on a quest and wandering aimlessly, and bring into the world a monk who can tell the player something about the fate of the missing parents.

Although the story generation focuses on the human players, there can be large-scale story-lines, which in turn affect the players. If a war breaks out, the player can suddenly find himself or herself in a battlefield. Also, the war can be incorporated to the player's story. Perhaps the parents' capture is due to the hostility between two nations—or perhaps the reason for it. As we can see from this short example, the story generator acts like a story-teller: it provides a sense purpose into the chaotic world.

### 3.4. Prediction and Production

The phenomena may concern states and events (i.e., time-dependent relationships). For example, a group formation can be identified by observing relative locations and orientations of soldiers close together. However, in order to recognize a forthcoming ambush, one must ponder on the maneuvers of the troopers.

Let us consider the temporal issues. Utilization of pattern recognition techniques in the case of time series data is illustrated in Figure 3. Here the world is seen as a generator that produces events and states, which can be named with symbols from the alphabet $\Sigma$. The current symbol sequence $S$ can be used to construct a model for the symbol stream (i.e., to learn the behavior of the generator). Modeling means recognizing the underlaying dependencies between the symbols. Typically the dependencies between the nearby symbols in the sequence are stronger than the distant ones. Therefore, it is often sufficient to consider a short term history, called modeling context, of length $\ell$ symbols. More accurate model can be achieved by increasing the length $\ell$ at the cost of time. There is a number of ways to model the sequence $S$, such as Markov chains and time series neural networks.

In Figure 3, the model gives the probability distribution for the next symbol in the sequence. This probability distribution can be used in two different purposes: *prediction* and *production* of forthcoming events and states.

When we are predicting the next symbol in the sequence $S$, we select the symbol with the highest probability in the model at the context $S$. Utilization of this prediction depends on the game. For example, in a fighting game the symbols could be movements like kick, hit, or duck. The generator is the opponent whose actions form the sequence $S$. Now we can predict the opponent's next move and prepare an appropriate counteraction.

The same model structure can be used to produce new events into the game world. For example, the actions of the computer controlled actors could be produced by applying the probability distribution of the model in context $S$. By selecting the next action randomly according to the probabilities of the actions, the behavior of the synthetic character could seem to be more reasonable. Obviously, this approach could be utilized in story generation that was discussed in previous section.

### 3.5. Other Employments

In addition to AI-related problems, pattern recognition can be utilized in finetuning the rules of the game world. For example, in RTS games we could observe that civilizations

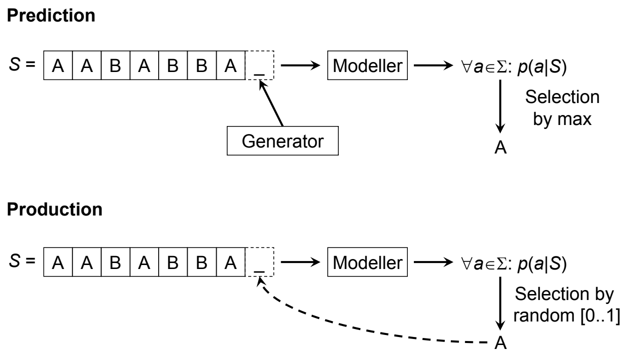

Figure 3: Model over a sequence of symbols can be used in prediction and production. Alphabet $\Sigma = \{\mathrm{A}, \mathrm{B}\}$.

are technologically unbalanced, or recognize the choke points both in the technology tree and in the terrain.

A good game has an intuitive interface that is easy to learn. The interface should adapt dynamically to the needs of a player. For example, in critical situations—which should be recognized— the player ought to be able to have more detailed control.

Likewise, living in a game world should be natural. Tutorials are a convenient method for illustrating the interface and the game world to the player. Hence, it can reduce the learning curve significantly. To be meaningful, tutorials should be adaptive to the skills of a player. This requires recognition of *modus operandi* of the player. This knowledge could be utilized also to keep the game challenging by proposing suitable difficulty levels for the player.

## 4. Discussion

Let us review Figure 1 in the light of CGs. The world represents now the game world, which exists only virtually in the computer. For this reason, CGs are in a slightly different position than most of the other pattern recognition applications. Whereas in other areas the developers can rely on the experts to help the development process, in CGs they are the experts themselves; the world is not given but generated by the very same persons. Also, we get so much information from the virtual environment that it is hard to discern which is useful for pattern recognition. For instance, in a RTS game the chosen strategy can be based on minutiae from the world but, in turn, that will slow down the pattern recognition process.

On the other hand, even in the computer games all patterns to be recognized do not originate from the virtual world itself. To implement challenging computer opponents we need to recognize the behavioral pattern of a human player, also. This is especially difficult due to human's ability to create extraordinary actions. Indeed, the lack of challenging and intelligent computer players is one of the main reasons for the popularity of the multiplayer games. Further, to provide a good game interface to a human player, we should observe the actions of the player and try to recognize the needs of information and interaction (i.e., relevant information is easily available and interaction with the game is smooth and intuitive). In fact, research results from augmented reality could be utilized in the development of user interfaces for CGs.

As we saw in Section 3.1, a pattern recognition system can be composed of several levels of detail. This could be utilized in design of difficulty levels for human players with varying skills. Because pattern recognition is separated on several levels, we can easily select how much information is provided to a higher level recognizer—even leave out some recognition parts. Similarly, different personalities of the synthetic players could be realized by filtering the recognized information.

Pattern recognition methods are widely used in CGs [6]. However, it is not always clear for the game developers where different pattern recognition methods are applicable. For example, the seems to be confusion where neural networks or fuzzy logic is best suited. Also, pattern recognition should be seen as reusable and modular software component whose correct design and implementation improve the overall efficiency of the game development process. Luckily, the gaming community is turning towards the academic community [4], and we—as members of the academia—should welcome this development with open arms. After all, we all want to have better and more entertaining CGs.

## 5. Conclusion

We introduced computer games as a quaint application area for pattern recognition. Pattern recognition is an important part of whole AI system by providing high-level information from the game world to the decision-making system. Thus, the task of pattern recognition is to extract relevant information from the world. We discussed how pattern recognition in CGs can be viewed from several perspectives.

## References

[1] M. Capps, P. McDowell, and M. Zyda. A future for entertainment-defense research collaboration. *IEEE Computer Graphics and Applications*, 21(1):37–43, 2001.

[2] G. Costikyan. I have no words & I must design: Toward a critical vocabulary for games. In F. Mäyrä (editor), *Computer Games and Digital Cultures Conference Proceedings*, pp. 9–33, Tampere, Finland, June 2002.

[3] Federation Internationale de Football Association. Laws of the game. Web page, Nov. 2002. http://www.fifa.com/refs/laws_E.html.

[4] International Game Developers Association. Web page, Nov. 2002. http://www.igda.org/.

[5] J.E. Laird and M. van Lent. Human-level AI's killer application: interactive computer games. *AI Magazine*, 22(2):15–25, 2001.

[6] S. Rabin, editor. *AI Game Programming Wisdom*. Charles River Media, Hingham, MA, 2002.

[7] RoboCup. Web page, Nov. 2002. http://www.robocup.org/.

[8] R. Schalkoff. *Pattern Recognition: Statistical, Structural and Neural Approaches*. John Wiley & Sons, New York, NY, 1992.