Intelligent User Interface Tools

Sameer Rajyaguru and Ergun M. Bicici and Robert St. Amant

Department of Computer Science
North Carolina State University
Raleigh, NC 27695
{srrajyag,embicici,stamant}@csc.ncsu.edu

Abstract. This paper describes an analysis tool for dynamic gaming environments. It is based on an image processing system and a contingency analysis tool for extracting the rules that the user is applying in a driving game simulation. The tool is intended for a cognitive modelling architecture that will learn from the user's previous actions to minimize the amount of assumptions made to control the environment. Interaction with the dynamically changing visual environments at real-time is important. This paper discusses our initial attempt to analyze dynamic gaming environments through statistical measures, our learning goal from previous experiences, the limitations, and its implications for user interface environments that dynamically change.

1 Introduction

Human-computer interaction experiences unbalanced talents of counterparts in the user interface, which can only be eased with the introduction of new solutions [1]. People take advantage of a rich verbal and nonverbal set of resources whereas machines have a set of sensors that map to commands and reactions. The resulting asymmetry limits the extent of the communication between humans and computers. Suchman [2] believes the solution can be provided by: (1) extending the access of computers to actions and circumstances of the user, (2) making the user aware of the computer's limits in accessing those interactional resources, and finally (3) compensating for the computer's inabilities with computational alternatives.

The designer of an interactive machine, as Suchman [2] calls, must ensure that the user gets proper response from the machine for his actions. Each interactive action assumes the intent of the actor with an adequate interpretation of the prior actions and the intent of the recipient with interpretation of the responses' implications. Thus, the interaction between computers and humans is dependent on each other's responses and their corresponding interpretations.

Interaction with the user interface of gaming environments are usually unpredictable and dynamic, which makes them hard to model. In this paper, we try to extend the access of computers to the user actions in a dynamic gaming environment, driving simulation, with an image processing system, and improve the learning abilities of computers with a statistical computation tool, namely contingency analysis.

This research is a progressive attempt on previous work [3]. We believe that statistical techniques may help cognitive modelers to grasp a better understanding of the

process of human reasoning. Our main goal is to automate user's tasks in the interface, such as driving a car, by analyzing the data that we get from both the user and the interface. Also, by providing a system for assessing what the user is trying to do, we intend to provide cognitive modelers with a wider range of tools with which their theories can be generated and tested. With this kind of help we may be able to minimize the amount of assumptions made to control the environment by cognitive modelers. The work in this paper describes early steps toward these two goals and emerge as another step toward automated exploration of user interfaces [4] that can guide and help the user. It can also serve as a computational medium for newly exploited ideas in user interfaces such as "programming by example" (PBE) [5] and "programming by demonstration" (PBD) [6]

In the sections below, we describe our approach. Our efforts are based on one of the three gaming environments mentioned in [3], a driving simulation.

2 Motivation

Our main goal is to automate interactions with the user interface by predicting user's intentions by observing previous actions and inputs to the computer. The speed of image processing systems has recently made real-time analysis of screen images feasible by a PBE system. [5] The basic claim of PBE is that you can program what you can see. Based on this claim, cognitive modelers can construct computer programs that simulate human behavior and reasoning in user interfaces. [3]

Cognitive models are basically computer programs that behave in the way that humans behave based on number of heuristic assumptions about "how humans do actually behave". In this sense, cognitive modelers try to gain insight into the process of human reasoning. [3]

A tool that would analyze the data from both the user and the interface will help cognitive modelers in generating rules and heuristics that the users are applying while interacting with the user interfaces. Such a tool will also help in automating the user tasks in an interface. Driving game provides us with a challenging environment to try these intriguing ideas.

3 Visual environments and User Interface Softbots

User interface softbots are intelligent software agents designed to control an interactive system through the graphical user interface. Previous detection efforts in softbots include using statistical pattern recognition techniques and rules and conventions in a Mac-based environment for finding the building blocks of the objects on the screen via a statistical search for more abundant forms [4].

We have worked with the 3D driver gaming environment, whose interface is shown in Figure 1. It is a first-person driving game, in which the user has control of the speed and steering of the car. The aim is to drive the car in the right lane and avoid accidents by staying on the road while avoiding obstacles such as other vehicles (e.g. cars, motorcycles). A detailed categorization of visual environments is given in [3]. According

to their analysis, the driving game that we focus fits into a dynamic, unpredictable, and complex category in a variety of ways.

Fig. 1. First-person driving game

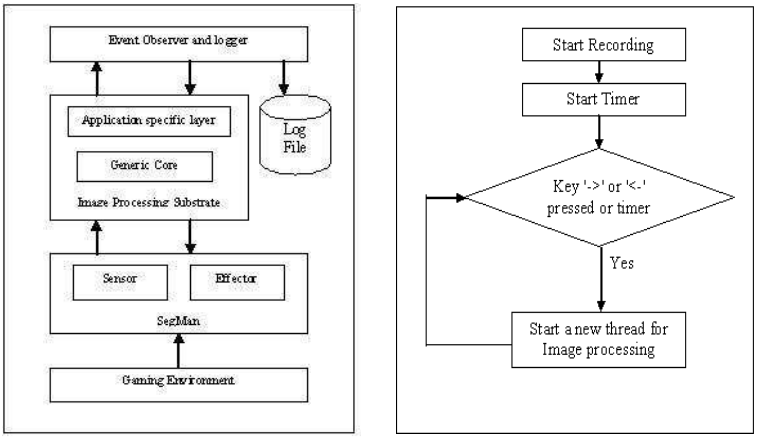
As the aim of the experiment is to come up with data that would assist in recognizing the rules humans employ while playing the game, we decided to log the coordinates of certain key points that users focus. The image processing system, currently, records the X-coordinate of the center of the game screen (i.e. the current position of the car), the X-Y coordinates of the midpoint of the right lane at the horizon and the X-Y coordinates of the midpoint of the right lane at some distance from the bottom of the screen (i.e. at some distance from the car).

For the driving game, the extensions are based on studies of human driving. Studies of driving behavior by Land and Lee [7] and Land and Horwood [8] describe a "double model" of steering, in which a region of the visual field relatively far away from the driver (about 4 degrees below the horizon) provides information about road curvature, while a closer region (7 degrees below the horizon) provides position-in-lane information. Attention to the visual field at an intermediate distance, 5.5 degrees below the horizon, provides a balance of this information, resulting in the best performance. Interested reader can take a look at the previous system for details [3].

3.1 Image processing system

As shown in the Figure 2, there are 4 distinct components that comprise the system. The various components are explained below:

– *Event Observer and Logger:* This component is tied to keyboard and timer events. It triggers the image processing routines to run in a new thread, based on event occurances and logs the data returned by the image processing routines. The keyboard events are the left arrow key and the right arrow key and the timer is set to 3 seconds.



a) High level architectural diagram

b) Control flow within the system

Fig. 2. System architecture

- *Image Processing Substrate:* This component captures and processes the game screen. It returns the values required to be logged, in order to better understand the user's responses to events in the game.
- *SegMan:* Segman is the component that the Image Processing Substrate uses to perform the low-level image processing tasks, like capturing the screen or extracting pixel information.
- *Gaming Environment:* The gaming environment has already been described in the previous section.

More details about the Image Processing Substrate and SegMan can be found in Shah et. al. [3]. The interaction with the interface objects of the game can be achieved via hooks that are provided in the development of the programming environment. The control flow within the system is shown in Figure 2 as well. A detailed analysis of the image processing routines can be found in Shah et. al. [3].

4 Algorithms for Data Analysis

In our user interface analysis system, we have a number of visual- and I/O-based sensors that result in different values for their measurements. These measurements can be either one of nominal, ordinal, or boolean such as the existence or absence of an attribute and our goal would be to make use of such cues that help a human cognizer reason.

Thus, given a set of streams of data, we can try to learn from it by looking at the occurrences of specific attributes. For example, if the number of cases where the user steers left when the road curves left occur more frequently than the cases where the user steers left when the road does not curve left or the road curves left when the user does not steer left, then we can say that steering left and curving left co-occur and are therefore associated.

This technique of finding the co-occurrence frequencies and calculating the strength of association between them is called contingency table analysis. A contingency table is a table of cross-classified data in which the attributes of each characteristic are matched with the attributes of other characteristics or their combinations to calculate the associations between them. In the case of matching two characteristics, this is called 2-way contingency table analysis. An example 2-way contingency table is given in Table 1.

Table 1. A 2-Way Contingency Table

	Road's Middle Point's X			
Steering Direction	220	236	195	*Total*
Left	5	1	2	8
Right	3	6	3	12
Total	8	7	5	20

Strength of association between two characteristics (2-way) are found with the following formula:

a: the number of cases they appear together
b$_1$**:** the number of cases where only first one appear
b$_2$**:** the number of cases where only second one appear

strength of association $= \frac{a \times a}{b_1 \times b_2}$

In the given table, the corresponding associations would be as follows:

Strength of Association (220, Left) $= \frac{5 \times 5}{8 \times 8} = 0.39$
Strength of Association (236, Right) $= \frac{6 \times 6}{12 \times 7} = 0.43$
Strength of Association (236, Left) $= \frac{1 \times 1}{8 \times 7} = 0.02$
. . .

For our purposes, we would like to generalize the problem to cover **N**-attributes and **M**-values in an **n**-dimensional attribute space. To do this, we need to extend the 2-way approach. Let's assume that we have **n** attributes that we acquire from our sensors and we want to look at the strength of association between those. Then, the general problem becomes selecting the subsets of attributes that appear together according to the combination subset and finding their associations by looking at the co-occurrences of those cases. We know that there are 2^n number of subsets for **n** attributes but not all of them have enough associative strength.

Finding the attributes that contribute to the relations among different data is a hard problem to solve since the number of possible subsets is exponential in size. Ockham's

razor is the sharp constraint that claims that the best theory is the simplest one that explains the data. Our contingency analysis program is based on this simplistic idea. To constrain the subset search space we use the minimum description length (MDL) principle, which is based on Ockham's axiom. If we think about the attributes in a contingency analysis table as the predicates that we can describe a system or the rules governing it, the suitability of the MDL principle becomes more obvious. By using only some number of attributes, we may reach the same or "good enough" representation ability. That's the reason why we seek to apply MDL principle to the problem of choosing subsets.

4.1 Contingency Table Analysis for N-attributed, M-valued Data

In machine learning, the common approach to contingency analysis is through a boolean attribute vector **A** of size **n** where methods to map from **I** instances of **A** to a boolean output are learned. We have extended this system to have an **N**-way contingency analysis with **M**-valued data for **I** instances. The complexity of such analysis is obviously intractable (i.e. non-polynomial in complexity).

The exponential growth in the number of possibilities reaches infinite fairly quickly with continuous data. In the case of the computer screen, even though we know that it is composed of finite number of pixels, it still poses important computational complexity for such an analysis.

Enumerating the subsets of a set with **n** elements is usually done by lexicographic ordering, which counts the subsets starting from the empty set. A lexicographic sequence for the subsets of a set of 3 elements is given in Table 2.

Table 2. Lexicographic ordering of subsets

Sequence	Subsets of $\{\diamondsuit, \heartsuit, \spadesuit\}$
000	-
001	$\{\spadesuit\}$
010	$\{\heartsuit\}$
011	$\{\heartsuit, \spadesuit\}$
100	$\{\diamondsuit\}$
101	$\{\diamondsuit, \spadesuit\}$
110	$\{\diamondsuit, \heartsuit\}$
111	$\{\diamondsuit, \heartsuit, \spadesuit\}$

However, lexicographic ordering does not order the subsets according to the number of elements they contain. The element "100" is not found until half of the subsets are traversed. So, if we want to apply MDL principle, we actually need to order the subsets by the number of elements in each subset. Loughry et. al. [9] present an algorithm for

efficiently enumerating the subsets of a set in the order of increasing number of elements they contain. The Banker's algorithm [9] provides a reasonable cut-off position when we do not want any lengthy rules to be generated. So, applying the MDL principle is easy. The sequence that we get from Banker's algorithm is given in Table 3.

Table 3. Ordering of subsets according to Banker's algorithm

Sequence	Subsets of $\{\diamondsuit, \heartsuit, \spadesuit\}$
000	-
100	$\{\diamondsuit\}$
010	$\{\heartsuit\}$
001	$\{\spadesuit\}$
110	$\{\diamondsuit, \heartsuit\}$
101	$\{\diamondsuit, \spadesuit\}$
011	$\{\heartsuit, \spadesuit\}$
111	$\{\diamondsuit, \heartsuit, \spadesuit\}$

4.2 Computational complexity

Contingency analysis does a complete search in the possibilities. There are a number of "good-enough" solutions that are based on heuristics [examples].

Matching a known set of objects consistently to the objects we recognize on the screen is the same problem as finding the maximal clique of consistent labels in region matching problem experienced in computer vision, which is known to be NP-complete [1]. There are different techniques applied to cope with the complexity of this constraint satisfaction problem, such as relaxation labelling. The idea is if we can represent the previously known objects as a set of constraints, we can use relaxation labelling for further relaxing these constraints to match newly recognized ones. This reduces the computational complexity.

Loughry et. al. [9] also show that using Gray codes or lexicographic ordering is not suitable when looking for a minimum subset. Efficiently enumerating subsets is important for this reason since the problem becomes computationally intractable without ordering.

To prune some weak associations beforehand and decrease the amount of computation, we are using two thresholds during the analysis. We use minimum support and minimum confidence to invalidate the attributes that do not occur frequent enough in the whole data set. We have set their values to 0.005 and 0.008 correspondingly, which corresponds to 5 to 8 instances in a data set of size 1000.

We need to get enough evidence to come up with some strong association required for us to associate keystrokes to the user interface organization that the user is seeing. This limitation enforces the data size to be big enough for such associations to be apparent as well as noise in the data to be relatively small enough to discard.

5 Experiments

In order to test the efficacy of our tool in analysing dynamic environments, we asked a couple of users to play the game and recorded their actions as well as certain key points that we thought would be useful. We got around 500 data points from the users and analyzed them.

The resulting associations and the best twenty set of the rules corresponding to them is given in Figure 3. This is the case without any discredization of the data.

Assoc	Combination	C_X	X_5	Y_5	X_M	Y_M	X_M2	Y_M2	Key
0.67401236	0 0 0 0 1 0 0 1					225			0
0.67401236	1 0 0 0 0 0 0 1	198							0
0.45674831	1 0 0 0 1 0 0 1	198				225			0
0.16117217	1 0 0 0 0 0 0 1	198							-1
0.16117217	0 0 0 0 1 0 0 1					225			-1
0.15750916	0 0 0 0 1 0 0 1					225			1
0.15750916	1 0 0 0 0 0 0 1	198							1
0.15062724	0 0 1 0 0 0 0 1			148					0
0.15012407	0 0 1 0 0 0 0 1			199					0
0.15012407	0 0 0 0 0 0 1 1							284	0
0.14868034	0 0 0 0 0 0 1 1							286	0
0.14815794	0 0 0 0 0 0 1 1							259	0
0.1415426	0 0 1 0 0 0 0 1			202					0
0.12702806	0 0 1 0 0 0 1 1			199				284	0
0.12690026	0 0 0 0 0 1 0 1						0		0
0.10124126	0 0 1 0 0 0 1 1			148				259	0
0.09521956	0 0 1 0 0 0 1 1			202				286	0
0.07917888	0 0 0 1 0 0 0 1				187				0
0.07858861	0 0 0 0 0 1 0 1						0		1

Fig. 3. Association strength table according to contingency analysis

We have plotted the graphs of the value of X at two different scan lines. The corresponding graphs can be seen in Figure 4 and Figure 5.

From the two graphs we can conclude that X 5 degrees from the horizon is a better factor to consider while driving as opposed to X measured in the middle of the road (close to the car). In this experiment, steering direction of 1 corresponds to steering right, 0 corresponds to no steering, and -1 corresponds to steering left.

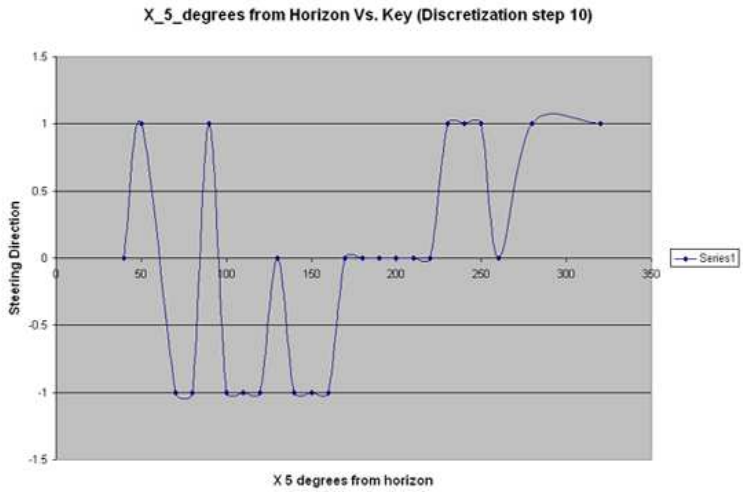


Fig. 4. Graph of the relationship between X 5 Degrees from Horizon and the steering direction

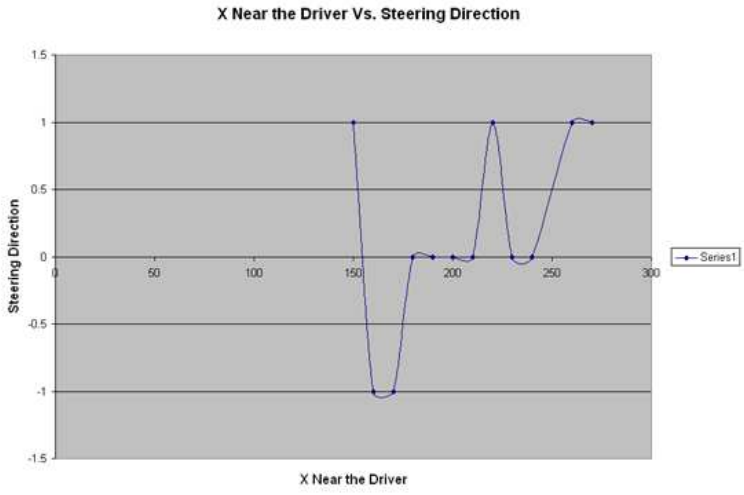


Fig. 5. Graph of the relationship between X middle and the steering direction

6 Conclusion

Our goal in building these systems is to provide cognitive modelers with a wider range of tools with which their theories can be generated and tested. In this system, we were able to find base components of driving environment. We plan to construct higher order components by studying the interactions between components.

We want to extend our analysis system to larger domains where we have more unknowns and more possible changes. For this, we are thinking of more complex gaming environments as well as other user interfaces where we can monitor user actions and automate their tasks.

Previous work [3] used cognitive models to control the system. Sending the rules to such a system with their corresponding confidence levels would enable a cognitive model to learn from the previous examples and the rules generated by them. An intelligent user interface helper would eventually need to learn from examples. Another possible work would be in robot control where we have many streams of unrelated data coming from different sensors. We can easily extend the possibilities into that domain as well.

References

1. Bicici, E.M.: Prolegomenon to commonsense reasoning in user interfaces (2002) (To appear).
2. Suchman, L.A.: Plans and situated actions: the problem of human-machine communication. Cambridge University Press, Cambridge, England (1987)
3. Shah, K., Rajyaguru, S., Amant, R.S., Ritter, F.E.: Image processing for cognitive models in dynamic gaming environments. In: Proceedings of the Fifth International Conference on Cognitive Modeling (ICCM-03). (2003)
4. Riedl, M.O., Amant, R.S.: Toward automated exploration of interactive systems. In Gil, Y., Leake, D.B., eds.: Proceedings of the 2002 International Conference on Intelligent User Interfaces (IUI-02), New York, ACM Press (2002) 135–142
5. Amant, R.S., Lieberman, H., Potter, R., Zettlemoyer, L.: Programming by example: visual generalization in programming by example. Communications of the ACM **43** (2000) 107–114
6. Cypher, A., Halbert, D.C., Kurlander, D., Lieberman, H., Maulsby, D., Myers, B.A., Turransky, A., eds.: Watch What I Do: Programming by Demonstration. MIT Press, Cambridge, MA (1993)
7. Land, M., Lee, D.: Where we look when we steer. Nature **369** (1994) 742–744
8. Land, M., Horwood, J.: Which parts of the road guide steering? Nature **377** (1995) 339–340
9. Loughry, J., van Hemert, J., Schoofs, L.: Efficiently enumerating the subsets of a set. (2000)