Understanding Human-Computer Interactions in Map Revision[⋆]

Jun Zhou, Walter F. Bischof, and Terry Caelli

Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2E8
{jzhou, wfb, tcaelli}@cs.ualberta.ca

Abstract. Tracking, parsing and modeling human-computer interactions to edit, revise documents is difficult but necessary if we are to develop automated technologies that will aid or replace humans. This paper introduces a system for accessing and recording a stream of events related to human actions in a real-time map (cartographic) revision system. The recorded events are parsed into a sequence of meaningful user actions and an action representation in XML format is generated. Further, we report results of experiments on predicting user actions such as view changes, edits, road tracking/production using a Hidden Markov Model.

1 Introduction

Analyzing and understanding human behaviour in software systems has gained increasing interest in Artificial Intelligence, Pattern Recognition, Human - Computer Interaction, and Cognitive Psychology. Surprisingly, very few research results have been utilized in real-world applications [1, 2]. A typical example is that of semi-automatic road tracking in images where the human is only used to initialize automated processes and as an editor at the end, with only a few or no human-computer interactions along the way [3]. Because of the deficiencies of the computer vision algorithms, the performance of the road tracking systems is very limited, usually not reliable, and less efficient than humans performing the task alone.

Here we have adopted a different paradigm: We study and model human performance on such tasks, identify key actions and difficulties, and then develop algorithms that improve the efficiency and accuracy of human operators. To achieve this, we need to track, parse and model the user actions in real world tasks, something that is rarely done in the document processing area [4, 5].

In this paper, we introduce a real world application for computer aided map revision. The software environment and the development tools in this project make it easy to keep track of both the temporal and spatial information of the

⋆ This project is funded by a grant from the Natural Sciences and Engineering Research Council of Canada

user actions in system-level events. These events are parsed into time-segments, meaningful and complete user action data stored in XML format. These data can be used to model user behavior patterns, and to help and automate the map revision process. In the next section, we briefly introduce the Raster Graph Revision (RGR) system in the United States Geological Survey (USGS), the class of maps of interest in this work. In Section 3 and 4, we present a system for tracking and parsing user actions. In Section 5 we present and discuss the experimental results in modeling human viewing behaviour and evaluate user performance using hidden Markov models (HMMs).

2 RGR system

One of the main paper products of the USGS topographic maps for the USA is 7.5-minute quadrangle topographic map, which is the only uniform map series that covers the entire area of the continental United States in considerable detail [6]. This map series consists of about 53,000 map sheets.

Current USGS maps are printed on white paper with six colors: black, red, brown, green, blue and purple, one for each feature. The RGR system uses existing film separates as the primary input and creates new film separates as the primary output. Cronapaque positives are produced photographically from the map separates and are scanned at 1000 dpi as raster images. The images, in addition to the digital orthophoto quads (DOQs) of the area to be mapped, are registered to the control file and displayed simultaneously on a computer screen as the source for revision. DOQs are orthogonally rectified images from aerial photos taken at height of $20,000$ feet with approximate scale of $1 : 40,000$. DOQ can distinguish ground objects of 1 meter, which is enough for ground object detection. Cartographers then make a visual comparison of the raster image and DOQs. When a discrepancy is found between a feature on the raster image and the DOQ, cartographers can add to, delete from, or modify the raster image to match the DOQ. Figure 1 shows the environment of the RGR system.

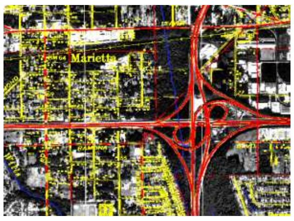

Fig. 1. Map revision environment. Here previous map layers are aligned with current digital image data.

The standard CAD tool for RGR systems is a Bentley Microstation. Bentley I/RAS B is used to display and manipulate the scanned map layers, Z/I Imaging

I/RAS C is used to display the DOQs, USGS RGR software provides CAD tools to draw, delete and modify specialized graphic symbols on maps, and MVES converts vectors and points to a symbolized raster format.

3 Tracking User Actions

In Microstation, interaction with the system is by keyboard or mouse. A simple drawing operation may be achieved by clicking a tool icon on the tool bar or by inputting a "key-in" command. To facilitate map revision, RGR uses a set of tools for cartographic symbols, each of which, along with mouse actions, encompass a sequence of system events (key-ins). Each key-in is considered as an event. Events from both inside or outside Microstation are processed by an input handler and are sent to an input queue where a task ID is assigned to each event.

With the imbedded Microstation Development Language environment, we can keep track of the states of the event queue and extract detailed information on each event, as described in Table 1. These time-stamped sequences of system-level events contain inter-action and intra-action information. The task, then, is to parse these sequences into meaningful higher-level user action sequences.

Table 1. Data structure for system-level event

event ID	ID of the event
event name	the key-in command
event type	Is it a keyin, coordinate, or reset?
event time	the time when the event is captured
event source	where does the event come from
x coordinate	x coordinate of the mouse clicking
y coordinate	y coordinate of the mouse clicking

4 Analyzing and Parsing User Actions

Altogether there are 278 tools in the RGR software, each corresponding to a human action. This number could be increased when new standards are adopted in the USGS. Analysis of all these tools is not necessary. First, some tools are used for features that rarely appear, or need not be revised in most cases. Second, some tools relate to registration of the scans and DOQs, environment setting, file input/output, and map plotting, and are not involved in the feature collection process. Third, we only process actions related to feature collection, at least at this initial stage of the project. As a result the number of tools is reduced to 144, each being composed of a sequence of events. A complete action may contain a tool selection, a sequence of coordinate clicks, and a reset operation. View changes may occur before and after the coordinate clicks.

The actions are grouped into 17 groups, each action group being defined by the number of permissible coordinates and occurrence of reset operations. Groups G0 to G11 contain system setting and drawing actions (e.g. draw a class-1 road or undo a previous action), groups V1 to V2 contain viewing actions (e.g. zoom-in or pan-view), group R contains the ”reset” action, and group CO correspond to a coordinate click not involved in any action.

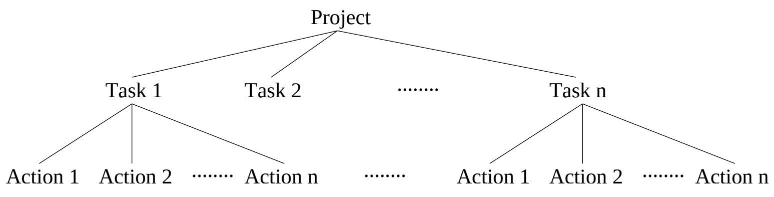

Fig. 2. Hierarchy of the action database

A sequence of system-level events is parsed into a tree structure as shown in Figure 2. The root of the tree is a project, which is defined as the revision of a map. A task is defined as the revision of specific ground object (e.g. a road, a block of buildings, or a lake). Semantic information is contained at both project and task level and can be tagged by human input. Finally, the user actions are stored in a XML format database.

5 Issues in Computer Aided Map Revision

The purpose of computer aided map revision is to improve both the speed and accuracy of the revision process. The solution to the first task is to reduce human involvement in the drawing task by using computer vision-based feature tracking. Until now, all research efforts fall in this area. However, the efficiency and accuracy of the computer vision algorithms are not necessarily consistent with human performance. This suggests that humans should be part of the feature tracking process. On the other hand, humans are not always accurate. Consequently a tightly coupled, real-time, error-correcting interaction between human and machine are what we envisage to make map revision more efficient. To realize this we need to better understand these interactions.

5.1 Predicting Human Gaze with Hidden Markov Models (HMM)

One of the ways to reduce human workload in map revision is to reduce drawing actions and viewing changes in the map revision process. This can be achieved by predicting when humans are likely to change viewing. To do so one can use Markov and Hidden Markov Models [7, 1].

Along with the parsed human action sequences, we can also record the sequence of viewing locations ("gaze"). We have performed several experiments with HMMs in order to study such viewing patterns. The states in the HMM were defined as groups of actions where the groups were defined syntactically and semantically. The syntactic groups were the 17 groups (17 states) defined in Section 4, and the semantic groups were obtained by clustering actions based on their functions as used in the RGR system, such as a group of actions for drawing transportation symbols or water body symbols. In this case, the actions were divided into 6 groups. The observations were calculated from the movements of the mouse. These movements were classified into either 9 (45^o step) or 17 (22.5^o step) directions, with one direction in each group being used for the no-movement cases.

Two participants were required to perform three drawing tasks twice, one for training and another for testing. The tasks involved the modification of roads, buildings, water bodies, etc. The average time taken to finish each task was 46 minutes. Altogether 34644 system-level events with time-stamps were captured and 9025 of these events were coordinate moves (changes in gaze). These events were parsed into 2157 actions. By average, each task sequence contained 180 actions. These task sequences were further cut into shorter sequences with 2 actions each. Finally, we obtained 560 training sequences and 540 test sequences sampled at seconds with an average length of 27 observations.

With the given number of states and observations, the degrees to which each observation sequence could be predicted from the trained and untrained models was determined by the degree to which the HMM could reproduce the movements over a number of Monte Carlo trials. The results of these experiments are shown in Table 2.

Table 2. The results of predicting next viewing change as a function of different number of states (S) and observations (O) for two training sets and two test sets. Values correspond to probabilities of correctly predicting the observation sequences given the model and the Viterbi MAP solutions. The right column shows chance performance levels. Average length of the observation sequences was 27.

	Training1	Training2	Testing1	Testing2	Chance
17S 17O	0.72	0.63	0.61	0.63	0.06
6S 9O	0.72	0.63	0.61	0.63	0.11
2S 9O	0.79	0.69	0.67	0.69	0.11

The numbers in Table 2 refer to the average probabilities of correctly predicting the observation sequences given the model and the Viterbi MAP solution over 100 Monte Carlo trials. This reduces to sampling from the state-dependent observation vectors given the Viterbi-predicted state for each observation value. Three models were tested. Each model was estimated by first obtaining initial estimates from the training sequences as all observations were automatically labeled (states known). The model was then updated via the Baum Welch algorithm. We also tested a two-state model in which the states were either system setting actions or drawing actions. The results show that in each state-observation combination, probabilities of correctly predicting the observation sequences, given the models and Viterbi solutions, were significantly above chance (right column of Table 2). These results are quite informative given the lengths of the sequences (27 in average) and prediction rates significantly above chance.

In real applications, predictions of viewing changes need to be much more accurate. Further, complete viewing prediction requires not only a direction, but also exact coordinates on the map. It suggests that simple analysis of the user actions is not sufficient. The prediction model should involve the recognition of features from image and maps, as described below.

Current research results in semi-automatic or fully automatic road tracking systems can be combined into the above model in order to provide support for complete viewing prediction and automatic tracking of roads. In automatic road tracking systems, the road seeds are found by the system without need to pre-select points along the road [8]. It is normally difficult to extend these automatic methods robustly and efficiently to very large images, such as the DOQs in this project [9]. Our viewing prediction result provide the possibility to reduce the searching area in a large image to relatively small areas in the predicted directions so as to generate a semi-automatic road tracker. How large the predicted area is can be decided by calculating the average length of the human viewing change steps. In the next subsection, we introduce a simple semi-automatic road tracking system on the assumption that target small area image has been extracted. Then we compare the performance of human and computer in road tracking.

5.2 Comparing the Performance of Human and Computer Vision-based Road Tracking

In semi-automated road tracking systems, a human operator typically provides initial parameters, such as an initial point, direction and road width [8]. Baumartner et al. implemented a system with more human computer interactions [10]. When the system detects a possible failure of the tracking module, it stops the system to allow the operator to select from a list of options. In all these semi-automatic systems, an assumption is made that the human can perform the task correctly and precisely. Further, what the computer determines as "incorrect" is also unclear and subject to error. This is not necessarily the case. To analyze this, we developed a simple road tracker and compared its performance with that of humans.

Note that a road segment is determined by two consecutive coordinates from mouse clicking and joining such coordinates defines the human detected road (axis). To compare this axis with that detected by computer, we cropped the neighborhood image of this road segment from the DOQ to reduce the search area (the size of a DOQ is more than 2MB) and then performed Canny edge detection [11]. As a result, points at maxima of gradient magnitude in the gradient direction are marked as edges, which may include both road and non-road edges, such as contour of cars. This edge operator is used because both straight and curved roadsides can be detected. Dramatic greylevel changes caused by surface material changes can also be detected and do not affect the extraction of candidate road edge points. Figure 3(a) and 3(b) show an example of the cropped image and the image after Canny edge detection.

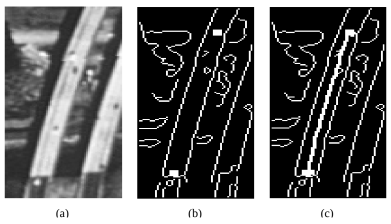

Fig. 3. (a) Cropped image from DOQ. (b) Human input (white blocks) and edges detected by Canny edge operator. (c) Axis detected by computer.

From each point on the axis defined by human, we constructed a line perpendicular to the axis and determined the intersection points to the edges. These points were the candidates of roadside. To reduce the disturbances on the road, like cars and shadows, points on the edges with short length were removed from candidate list. If two or more candidates were found, the road width limits defined by USGS was used as the upper and lower bounds of the distances between roadsides [12]. The two closest intersections to the axis detected by human, which also met the width limit, were selected as the roadsides corresponding to the axis point. Connection of the mid-points of these intersection pairs formed the axis detected by the computer. Figure 3(c) shows the result of Canny-edge axis detection of the image in Figure 3(a).

Two kinds of errors occurred during Canny-edge axis detection. One kind was caused by deficiencies in the Canny edge detection. When the road and the background have similar greylevels, Canny edge operator failed to mark the road edges. Therefore, part of one or both roadsides were missing. To reduce this error, the roadsides can be predicted by fitting a parabola to the most recent road points, as described in [13]. Another type of error comes from disturbances on the road that have not been removed. Some of them are connected with the roadsides, which made the road to be thinner than the lower bound of the road width limit. Consequently, correct road side candidates could not be selected by the system. This kind of error could be avoided by jumping to the next axis point along the road.

To compare the human and computer performance, the mean distance of the axis detected by computer and by human were calculated. Figure 4 shows the distribution of the distances versus road angle changes on two training sets described in the above sub-section. The road angle change was obtained from the angle between two consecutive axis detected by human. Originally, we expected

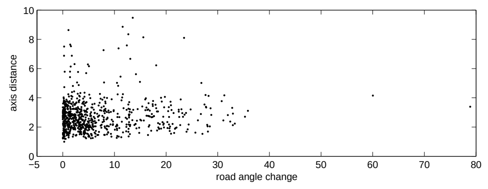

Fig. 4. Distances of road axis versus road angle changes on two training sets.

that the distances between the axis would increase along with the road angle changes. However, the results shows there is no explicit relationship between them. In most cases, the distances between the axis were less than 4 pixels despite the change of road angles. It is within the tolerance of positional accuracy defined by USGS (maximum 6 pixels, average 3 pixels) [14]. In the cases where the distances were too large to be acceptable, analysis shows that although most errors were caused by the deficiency of the Canny-edge axis detection, some are caused by the inaccuracy of human in georeferencing an ground object in DOQ with a map feature. The human input may lies much closer to one roadside, or even falls outside of the road. In these cases, the road tracking system may not select the correct roadside candidates. An example of this deviation is shown in Figure 5.

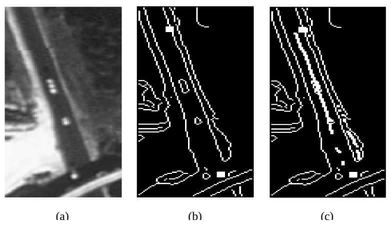

Fig. 5. (a) Cropped image from DOQ. (b) Human input and edges detected by Canny edge operator. (c) The white blocks at the end of the road are the input from human. The small white dots show the axis detected by computer. Because the human input road end points are shifted from the true centers, the computer can not detect all the axis points correctly.

6 Conclusion

This paper has introduced a real world environment for map revision. The user actions in the map revision are tracked and recorded as a time-stamped sequence of events in the system level. Actions are classified into groups according to their usage pattern, and a parser is used to parse the event sequence and generate a XML format user action database. This is the first open database on user behavior in real world applications involving document processing, feature tracking, pattern recognition that has been reported.

Two experiments based on the human data are reported in predicting human viewing changes and comparison of the performances of human and computer in road tracking. It is clear from these studies that in order to build a human-machine system capable of improving human performance, we need a more tightly coupled interaction between human and machine.

References

1. Horvitz, E., Kadie, C., Paek, T., Hovel, D.: Models of attention in computing and communication: From principles to applications. Communications of the ACM **46** (2003) 52–59
2. Encarnacao, M., Stoev, S.: An application independent intelligent user support system exploiting action-sequence based user modeling. In: Proceedings of the Seventh International Conference on User Modeling, Banff, Canada (1999) 245–254
3. Baltsavias, E., Hahn, M.: Integrating spatial information and image analysis one plus one makes ten. In: International Archives of Photogrametry and Remote Sensing. Volume 33. (2000) 63–74
4. Sandanayake, P.T., Cook, D.J.: ONASI: Online agent modeling using a scalable Markov model. International Journal of Pattern Recognition and Artificial Intelligence **17** (2003) 757–779
5. Mayfield, J.: Controlling inference in plan recognition. User Modeling and User-Adapted Interaction **2** (1992) 83–115
6. Moore, L.: The USGS Geological Survey's revision program for 7.5-minute topographic maps. http://mcmcweb.er.usgs.gov/topomaps/revision.html (2000)
7. Hacisalihzade, S.S., Stark, L.W., Allen, J.S.: Visual perception and sequences of eye movement fixation: a stochastic modeling approach. IEEE Transactions on Systems, Man, and Cybernetics **22** (1992) 474–481
8. Fortier, M., Ziou, D., Armenakis, C., Wang, S.: Survey of work on road extraction in aerial and satellite images. Technical report, Université de Sherbrooke (2000)
9. Geman, D., Jedynak, B.: An active testing model for tracking roads in satellite images. IEEE Transactions on Pattern Analysis and Machine Intelligence **18** (1996) 1–14
10. Baumgartner, A., Hinz, S., Wiedemann, C.: Efficient methods and interfaces for road tracking. In: PCV02. (2002) B: 28
11. Canny, J.: A computational approach to edge detection. IEEE Transaction on Pattern Analysis and Machine Intelligence **8** (1986) 679–698
12. U.S. Geological Survey, U.S. Department of the Interior: Standards for 1:24000-Scale Digital Line Graphs and Quadrangle Maps. (1996)
13. Jr., D.M., Denlinger, J.: Cooperative methods for road tracking in aerial imagery. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Ann Arbor, Michigan (1988) 662–672
14. U.S. Geological Survey, U.S. Department of the Interior: Standards for Raster Feature Separates Version 1.0. (2002)