University of Alberta

A Human-Computer Interaction Framework for Image Interpretation in Cartographic Map Revision

by

Jun Zhou 

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Department of Computing Science

Edmonton, Alberta
Fall 2006

Your file Votre référence
ISBN: 978-0-494-23138-8
Our file Notre référence
ISBN: 978-0-494-23138-8

Abstract

Over the past decades, considerable progress has been made in the area of automatic image interpretation using computer vision and pattern recognition methods. However, there is still a large gap between the requirements of most image interpretation applications and the accuracy and reliability achieved by automatic methods. Many attempts to automate these tasks are too fragile, and they require checking by experts before any final decision can be made. For this reason, most successful systems retain a "human in the loop", where a human operator can aid the automatic image interpretation through human-computer interactions (HCI).

In this thesis, we introduce a framework for image interpretation based on HCI. This framework consists of five components, a human-computer interface, a user model, computational image interpretation models, a knowledge transfer scheme, and a performance evaluation scheme. We applied this framework to image feature tracking in cartographic map revision, which is expensive, time-consuming, and currently has to be done manually. We implemented an interface to access, record, and parse human actions. The human data was used to predict user actions (such as view changes) using hidden Markov models and to develop a computer-assisted road tracking system. In this system, the human operator retains complete control over the operations with the computer acting as an apprentice and, later, as an assistant. The apprentice learns simple tasks from the human operator by tracking and modelling all input operations in road tracking. Eventually the apprentice can take over these tasks from the human and execute them, returning control to the human operator whenever problems arise. Two tracking methods were implemented, using Bayesian filtering and profile matching, as well as online learning and novelty detection. Experimental results confirmed that our approach is effective and superior to existing methods. Our approach is computationally efficient, and it can rapidly adapt to dynamic situations where the image feature distributions change.

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor Dr. Walter F. Bischof. I can't finish this work and my thesis without his guidance, encouragement, patience and support. I will never forget the days that we spent together, discussing algorithms, methods, and life.

I am also grateful to my former co-supervisor Dr. Terry Caelli, for his guidance, encouragement and help throughout my research. He suggested this topic, and supported my research in all aspects. The time and effort he dedicated to this work is valuable.

Special thanks to Dr. Arturo Sanchez-Azofeifa, who taught me a lot in GIS and remote sensing, and Dr. Li Cheng, who worked on and discussed with me many research topics.

I am thankful to Lawrence L. Jontz, Charles E.Conley, and Deborah E. Cochran in the Mid-Continent Centre of the United States Geological Survey, for their tutorials on the map revision process and systems, as well as warm host when I visited Rolla, MI. Thanks to William Tompkinson, Clare Davies, David Hollan, and Nick Groome in the Ordnance Survey of the United Kingdom, for their participation in the map revision project and discussions.

I am also indebted to Frances Moore, Edith Drummond, and Karen Berg, for their help in my Ph.D. studies and research. Many thanks go to my friends in University of Alberta for their fruitful discussions with me and their friendship.

I can not express my gratitude with words for the love and support from my wife Zhichun Fu, who spent a lot of time and effort taking care of my family and me. I also like to thank my parents Guanghua Zhou, Weixia Zhou, and my sister, Ling Zhou. The care and encouragement from my family are the power to push me forward and will accompany me forever. Finally, special love to my son, little Eric F. Zhou, who brought me a lot of trouble, but more happiness, in the past two and half years.

This research has been supported by fellowships from NSERC, iCORE, and School of Graduate Studies at University of Alberta. Bentley Systems Inc., Intergraph Corporation, and BAE Systems provided supports on software systems.

To my family

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Automatic image interpretation has a wide variety of applications, for example, image retrieval, scene understanding, medical image analysis and remote sensing. Compared to human image interpretation, automatic methods are fast, mathematically well defined, and suitable for the processing of large volumes of data. For this reason, they have been studied extensively by researchers in Computer Vision, Pattern Recognition, Machine Learning and Remote Sensing. However, even with the rapid progress in this area, there is still a large gap between the requirements of most image interpretation applications and the accuracy and reliability achieved by the automatic methods. For example, it is hard to completely and accurately detect roads in a satellite image. In image data mining, correct image classification is still a difficult task. In many real-world applications, such as medical image processing, bank cheque processing and map production, mis-interpretation of image features may even generate legal problems. It is thus no wonder that in these applications, image interpretation tasks are still performed manually. The disadvantage of manual work is that it is slow and expensive. With the exploding amounts of available image data and the increasing demands on data analysis, total reliance on human image interpretation is not feasible in most industries.

One solution to this problem is to further study the ability of humans to perceive the world, and to acquire knowledge in order to build better automatic image interpretation algorithms. Efforts have been made in Cognitive Science, Computer Science, and Neuroscience, focusing on modelling how humans learn from data. It is expected that the modelling will improve the accuracy and robustness of a given learning algorithm, in order to build a better prediction model before it can process unseen data automatically.

As an alternative to building completely automatic systems is to retain a "human in

the loop", where a human operator can work with an automatic image interpretation system through human-computer interactions (HCI) [31, 28, 77, 21]. This is particularly in the area of HCI, where it is widely believed that computer vision-based user interfaces will be one of the major forces in the future developments of HCI [101, 110, 94]. Some successful systems adopted the human-in-the-loop solution because many attempts to completely automate image interpretation systems are too fragile and require checking by experts before any final decision can be made. This solution has several advantages in real world applications. First, no optimal prediction model is needed at the initial stage of the image interpretation, because human-computer interactions provide the possibility of improving the prediction model. Second, it provides users with the choice to switch between automatic and manual methods so as to gain better control of the image interpretation process and to build more usable systems. Third, a number of prediction models can be implemented, so that they can boost each other's performance. Further, users can select a better prediction model when multiple models are available. Fourth, it is possible to build user-adaptive prediction models by monitoring the performance of the semi-automatic systems. Finally, whenever errors happen during the prediction, the human can get involved to correct them quickly.

Although the concept of building human-computer interaction systems has been widely accepted in the research and industry, studies in this area are quite biased. Current research focuses on two aspects, the human side and the computer side. The former tends to provide human with better interfaces by developing hardware or software tools, and by modelling user behavior, while the latter tends to replace human from the manual work by developing computational algorithms and models. Although great achievements have been made in these two aspects independently, it is gradually clear that they are closely related with each other. As a consequence, many semi-automatic systems have been proposed in the past a few years, attempting to combine both human and computer resources.

In this thesis, we attempt to bridge the gap between the human and the computer: we study and model human performance on image interpretation tasks, we identify key actions and difficulties, and we then develop algorithms that improve the efficiency of human operators. At the same time, we explore how interactions can improve the performance of automatic systems through knowledge accumulation and online machine learning. The optimization of the human and computer resources requires the development of control strategies that can adapt to a variety of perceptual tasks such as algorithm selection, feature extraction, and training/testing data retrieval. For this purpose, we propose a robust framework that effectively combines existing technology with task demands and human

performance. It is a practical solution to the applications in image interpretation, where many automatic algorithms have been developed, but most of them have been unusable in reality [43].

1.2 Application Background in Cartographic Map Revision

With the creation of the first map, the tedious but non-trivial task of map revision began as well. For hundreds of years, the revision process has stored maps for information that humanity needs, and has helped us to understand the world that we are living in [114]. Traditionally, map revision tasks were performed manually. Departing from analog cartography, cartographers now perform this task in a digital environment. The tools used by users are no longer pencils and plastic films, but computers, databases and printers. CAD environments like Microstation and Adobe Illustrator facilitate users in designing and editing maps. GIS systems like GeoMedia and Arcinfo make it possible to manage different geospatial databases and integrate different formats of data into one system. With the development of remote sensing technology, aerial images and satellite images have been widely used in map revision to present the latest ground truth on earth's surface.

We study cartographic map revision because it has a well defined context to examine our ideas. The software environment and the development tools make it possible to keep track of the user. It enables us to record the spatial and temporal information of the user actions, as well as to analyze and model them. Another advantage is that the revision tasks are well defined. Normally, revisions are performed on multiple map layers, which contain map features such as boundaries, road, building, water bodies, and others. Many automatic methods have been developed to detect and track these features in remote sensing images, using automatic image interpretation technology. We are focusing on the technology to interpret these features, utilizing and extending the existing automatic methods, and studying their interaction mechanism with users. Then we can compare our approach to automatic methods and human performance, respectively.

In this thesis, we present our work on semi-automatic road tracking, one of the key tasks in map revision, using the proposed HCI framework. We developed a system to parse system-level software events into time-segmented, meaningful and complete user action data. These data can then be used to analyze user behavior patterns in road tracking. We also propose a human-centered approach for road tracking, in which the human works as a tutor and decision maker, with the computer acting as an apprentice and, after training, as an

assistant. The computer tracks and models human actions, and, on request, takes over those tracking tasks that can reduce the human effort. Our approach is computationally efficient, and it can rapidly adapt to dynamic situations where the image feature distributions change.

1.3 Outline of Thesis

This thesis is organized in nine chapters.

In Chapter 2, we review the state of the art of semi-automatic image interpretation and computer-aided map revision. This covers a variety of topics, including HCI techniques in image interpretation and analysis, semi-automatic image interpretation methods for map feature detection and tracking.

In Chapter 3, we propose a general HCI framework for semi-automatic image interpretation. This framework consists of five components, a human-computer interface, a user model, a computer vision component, a knowledge transfer scheme, and a performance evaluation scheme.

In Chapter 4, we first review current map revision programs. Then we apply the HCI framework to map revision. The software environment and the development tools make it easy to keep track of the user actions in system-level events. These events are parsed into time-segmented, meaningful and complete user action data that are stored in XML format.

In Chapter 5, we show how the parsed user action data is used to analyze user behaviors. Two experiments were performed. First, user actions were used in modelling human viewing patterns using hidden Markov models (HMMs). This included a training process and a testing process. Second, user data were used to compare to automatic road edge detection method.

In Chapter 6, we introduce a linear feature extraction algorithm to extract lines from noisy aerial images. This algorithm uses directional information to filter an image and to extract edges. Then, the Hough transform is used to compute peaks from the edges and to restore line segments. A peak selection algorithm based on directional information is used to distinguish true line segments from the line segments generated from spurious noisy edges.

Applying the HCI framework to road tracking tasks, a semi-automatic Bayesian filtering method is introduced in Chapter 7. The Bayesian filters are used to estimate the current state of the system based on past and current observations. User input not only sets the initial state of the Bayesian filters but also reflects knowledge of road profiles. A knowledge base

is built to facilitate the knowledge transfer between human and computer. Experimental results confirm the effectiveness of this approach, and it is shown to be superior to existing methods.

In Chapter 8, we propose an online learning approach that naturally integrates guidance from human experts with automatic computer vision algorithms for tracking roads in aerial photos. Human inputs provide the online learner with training examples to generate road predictors. An ensemble of road predictors is learned incrementally from human inputs, and the predictors are then used to automatically track roads. The experimental results on the learning and tracking models are compared and analyzed. The proposed approach is computationally efficient, and it can rapidly adapt to dynamic situations where the road feature distributions change.

Finally, we summarize the thesis in Chapter 9, with some conclusion remarks and propose the future work.

1.4 Contribution of the Thesis

The ultimate goal of studying image interpretation is to aid humans in performing the task in a more convenient and efficient manner. Current research in semi-automatic image interpretation has not been very systematic. Researchers often adopt solutions that seem work even if they only have a rough idea on what to do and how to do it. They often miss why they do it in that way, and how to do it better.

The human-computer interaction framework introduced in this thesis is one of the first to analyze semi-automatic image interpretation systems in a systematic fashion. It can improve the efficiency of image interpretation in performing tasks usually done by human, while maintaining completeness, correctness, and accuracy, because the human is never removed from the process. This framework is very generic and could be applied to similar applications that require mutual understanding and different levels of interactions between human and computer. This is the most important contribution of the thesis.

We made an in-depth analysis of all components that are required for semi-automatic image analysis, with special attention being paid to the interaction mechanism between human and computers. We defined the roles of human and computer, showing that the key to bridge them is to build an effective knowledge transfer scheme. Our scheme enables both static and dynamic knowledge to be transferred from human to computer, so that the computational model can be gradually strengthened. This suggests that we can take advan-

tage of the current achievement in computational models, without the need to explore many more new methods, so that great efforts can be saved to make workable systems.

We demonstrated the application of the proposed framework in solving real-world problems. In designing a semi-automatic road tracking system for map revision, we first modelled human behavior in manual work, and identified why we can solve the problem using semi-automatic method, and how this should be done. Then we developed systems that enables human control and knowledge transfer through human-computer interactions. Experimental results confirm that our approach is effective and superior to existing methods. This is the practical contribution of the thesis.

Chapter 2

Semi-automatic Image Interpretation Methods and Map Revision

The term image interpretation, or image understanding, often has different meanings in different research communities. In the computer vision community, it refers to the reconstruction and interpretation of a scene by means of images. It covers the broad topics from early vision to high-level vision [15]. In the remote sensing community, it is defined as the extraction of qualitative and quantitative information from images. Thus, an image interpretation system consists of image acquisition, image processing, feature extraction, and image analysis [85]. In both communities, the common topics are image processing, feature extraction, and object recognition. In these topics, studies on human-computer interactions are one of the key directions, and they form the basis of developing semi-automatic systems. In this chapter, we review past work and put special emphasis on the HCI techniques for image interpretation and the state of the art in map feature detection.

2.1 Human-Computer Interaction for Image Interpretation

Human-computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems, and with the study of major phenomena associated with them [33]. Some interesting topics in HCI that relate to the image interpretation include human-computer interfaces, human information processing, and performance evaluation.

2.1.1 Human-computer Interface

A variety of aspects are important for a human-computer interface. In Graphical User Interfaces (GUI), such aspects include the presentation of visual components (desktop, windows,

menus, icons, and others), the way data are presented and operated on, and the supporting hardware. In our study, we are concerned with the way human actions are input into the image interpretation system. The input method determines directly how the input information can be captured and used in the later processing.

The most typical input devices in a GUI are keyboard and mouse. These two devices are suitable for two-dimensional (2D) environments that use dialogue techniques, such as text input, selections and discrete parameter specification. They provide support for fast, simple, convenient, and interactive continuous control in a system. Many image interpretation systems use keyboard and mouse as input devices, e.g. for contour editing [71, 97, 29], segmentation [37, 107], or object modelling and detection [103, 84, 151, 36]. Some interaction techniques have been proposed for mapping two-dimensional cursor motion into a three-dimensional (3D) translation or rotation [55, 57]. These techniques usually consist of a combination of mappings of planar movement into space and a mechanism for switching the control of the current degree-of-freedom, which are not trivial task.

It is more natural to use 3D input devices to support the 3D interactions. A number of 3D devices were compared in [57], including popular haptic devices such as 3D mouse and gloves. 3D mouse is popular input device in photogrammetry and medical image processing [139, 48]. It allows user to control the 3D position of a cursor. Functions keys are used to set parameters, change modes or perform other functions. Instrumented gloves are normally used in combination with sensors so that they can detect the hand position and finger movement [142].

Beyond the conventional user interfaces, advanced user interfaces take advantage of recent progress in pattern recognition and computer vision. These interfaces include systems for gesture, action, and speech recognition, which make it possible to interact with image interpretation system at a distance. Colombo *et al.* reported a system for image manipulation using hand pointing actions [22], which allows users to move freely in a room with an image displayed on a big screen. Users use their arms as the pointing device, and the movements of the arms and hands are tracked by a pair of cameras. The hand pointing gestures are mapped onto the screen locations of interest so that an subarea on the image can be retrieved. A similar system was reported by Arsenio to guide image segmentation for robot vision [3]. First, a standard color segmentation algorithm was applied to a stationary image. Then the human actor waved an arm to indicate the locations of the objects of interest. Finally, a robot mapped the object to target classes by matching the color cluster to predefined object templates. Speech recognition techniques are also used in image interpretation

Table 2.1: Summary of semi-automatic image interpretation systems

systems	applications	interfaces	usage of input	evaluation
Kass [71]	contour detection	mouse	initialization	qualitative
Elder [29]	contour editing	mouse	judgement	qualitative
Freedman [37]	image segmentation	mouse	preprocessing	qualitative
Harders [48]	image segmentation	3D mouse	steering	quantitative
Arnqvist [2]	image segmentation	keyboard, mouse	initialization	quantitative
Okada [106]	image segmentation	mouse	initialization	quantitative
Francois [36]	3D Modelling	mouse	initialization	qualitative
Lepetit [84]	object detection	mouse	preprocessing	qualitative
Giakoumis [42]	object detection	mouse	initialization	qualitative
Xiong [140]	object detection	N/A	correction	quantitative
Zou [151]	object recognition	mouse	initialization	quantitative
Gougeon [44]	object recognition	mouse	preprocessing	quantitative
Arsenio [3]	scene understanding	action	preprocessing	quantitative
Okabe [105]	document retrieval	mouse	feedback	quantitative
Muneesawang [99]	image retrieval	mouse	initialization	quantitative

to identify and tagging relevant objects and regions in the image. The Show&Tell system proposed by Srihari and Zhang is such an example [124]. A user views the image and describes it in spoken language to indicate the objects and regions in the image. The input is processed by a speech recognition engine to generate constraints on the image, such as the spatial relationship between objects. The user can also use the mouse to provide additional information. Then the image is processed by an automatic image interpretation system to extract objects based on the user inputs.

As summarized in Table 2.1, the mouse is the most popular input device for image interpretation. In many traditional image interpretation tasks, such as image retrieval, medical image processing and remote sensing, a user has to interact with images at a close distance. Such tasks require simpler, more accurate and more convenient interfaces, for which a mouse is the most suitable input device and will remain so in the near future.

2.1.2 Usage of Human Data

Once the human inputs are obtained from the interface, the next step is to process these inputs. Depending on the interactive type, human input can be parameterized and used in different ways. Depending on this interaction, semi-automatic image interpretation systems can be divided into user-assisted systems and computer vision-based interaction systems [97]. User-assisted systems describe techniques where the user interacts first in image space, so that the algorithm produces a desired result. The computer vision-based inter-

actions are the methods where the computer perform some or all vision tasks, before the human gets involved as decision maker.

User-assisted System

In a user-assisted system, the human provides inputs for two purposes, preprocessing and initialization.

- **Preprocessing**

  F. Gougeon reported a semi-automatic forest inventory system based on individual tree crown recognition, in which humans helped to remove the non-forest areas on the digitized high-resolution aerial photographs [44]. Then the automatic delineating algorithm was used to isolate the individual tree crowns and to generate species signatures. In contrast, in [3], humans were used to point out areas of interest in the image before the classification of the object in the area could be done. A similar HCI system was developed in [84] to outline the occluding objects in the key-views, before the actual boundary of the object were tracked by *snakes*. The medical image segmentation system proposed by Freedman and Zhang required users to mark seed pixels both within and outside of the object of interest [37]. Shape priors were then incorporated to help the automatic segmentation.

- **Initialization**

  In these systems, human inputs guide the setting of parameters for image processing [42, 151, 2, 106], feature extraction [141, 42], object detection [83, 56] and image classification [99, 88]. Some widely used algorithms also require humans to set the parameters. For example, in region-growing, the users point out one or more seeds [1, 141, 42] to initialize the segmentation. These seeds can be pixels or regions, and neighboring pixels or regions can be merged into the seeds. An advantage of human involvement is that the human can help to make judgements on ambiguous region. In active contour models, a set of control points is often input by the human [71, 13]. The snakes, which are splines with active shapes, are updated iteratively to minimizing their energy until the weighted internal and external energies reach a local minimum. In modelling 3D objects from 2D images using a multi-camera system, the human can help the computer to establish stereo and pose-to-pose correspondences, after the object features, such as edges, vertices, curves, and planes, have been extracted [98]. Francois and Medioni proposed a system to produce a 3D

model of an object as observed in a single image [36]. User were involved in the system with a minimum of high-level interaction for feature extraction and grouping to enable robust computer vision algorithms for geometric computing.

In summary, user-assisted techniques are suitable for systems that require user involvement before the automatic process. The purpose is to facilitate the analytical or numerical computation. The quality of the human inputs greatly affect the performance of the automatic processing. When an initial input is poor, the automatic processing might not generate correct or accurate results. In this case, iterative user inputs are required, and sometimes user have to provide feedbacks or correct errors, which can be considered as computer vision-based interactions.

Computer Vision-based System

In computer vision-based interaction systems, the purpose of human input is to judge the results, provide feedbacks, and correct errors for the computer vision algorithm.

- **Judgement**

  The human evaluates the automatic processing results and decides what to do in the next step. One example is the Interactive Contour Editing [29, 97], which is an image editing tool that supports the translation of actions from the contour domain into actions that will be performed on the image. The key step is to find reliable image contours, to encode their locations, and then reconstruct a high-fidelity representation of the original image. Image contours are detected automatically using edge detection methods, and the human operator observes the contours and decide how to group them. In another example, Navalpakkam and Itti studied how human attention was influenced by high level tasks [102], and designed an architecture to estimate the task-relevance of attended locations in a scene. Vision-based components worked together to estimate the task-relevance of the most attended areas. Then the object in the area was recognized by a human operator.

- **Feedback**

  Feedback can be used to adjust the behavior of the automatic systems. These feedbacks can be qualitative, indicating whether the results are right or wrong, or quantitative, indicating how good the results are. Sometimes, feedback can be considered as a human labelling process, which is especially useful in developing systems based on online machine learning. For example, Muneesawang *et al.* developed a labelling

system for content-based image retrieval [99]. In each image retrieval session, users were asked to classify the relevant and irrelevant results related to the query images. Then the classification results were used as training examples for the retrieval model. Similar work include the roof detection system by Maloof *et al.* [88] and the interactive document retrieval systems by Okabe and Yamada [105], by Salton and Buckley [118], and by Peng *et al.* [111].

- **Correction**

  Computer vision-based image interpretation is not robust, hence errors happen from time to time. When a computational method fails, the human is the best source of remedy. In the systems reported in [143] and [140], the human operator was required to check and correct the road detection results after automatic road extraction from high-resolution imagery and road database. Another example is the Crayons system that supports such interactions [32]: With the help of human input, an automatic object classifier generates initial results to distinguish an object from background. The human analyzes the results, and make new inputs to correct the classifications results. With the new inputs, the classifier also updates itself using simple machine learning methods, like nearest-neighbor algorithm, or decision trees.

Steering System

A complete semi-automatic system could be a combination of the user-assisted and vision-based models. Sometimes it is hard to distinguish between these two. A human input may be used both to provide feedback and to initialize an algorithm, as in the content-based image retrieval system in [99], where human inputs provide labels first and then are used to initialize the training process. We call such closely coupled interaction models steering models, in which human guides the computational process, gradually optimizing the algorithms until the desired results are achieved. Only a few steering models have been reported in the literature, including the process model by Madhok and Landgrebe [87], the satellite image processing system by Datcu and Seidel [25], the building extraction system by Gulch [47], and the image segmentation system by Harders and Székely [48].

In building their process model [87], Madhok and Kandgrebe proposed three design principles. First, human and computer have different abilities and thus need to take on different roles in a system. Second, the machine is used to validate the human hypotheses. Third, not every analysis can be solved perfectly. Thus, multiple techniques should be combined to generate successful results. These principles were applied to remote sensing

data analysis. Aerial images were segmented and classified to generate thematic maps. This output was validated with test data gathered from humans.

Datcu and Seidel proposed a knowledge-driven information mining system for remote sensing data processing based on human-centered concepts [25]. Input images were first processed by computer vision algorithms to extract primitive features such as texture and geometrical information. These features were clustered and used to organize a quicklook images archive. The human was required to give examples of target objects in the quicklook images, and examples were automatically generalized with confidence values for all images in the archive. Low confidence value objects were verified by human. This interaction system was implemented to generate risk maps for mined areas and to detect objects and structures in high-resolution images.

In [48], Harders and Székely introduced an interaction tool for medical segmentation. Human input was required to generate the centerline of a tubular structure in 3-D data using haptically assisted tools. Then the centerline was used to initialize a deformable surface model. This model was used to segment the medical images. A step-by-step segmentation approach was used to hide the already segmented image parts, thus, the human was in control of the whole process. The author claimed that the using of multimodal interface improved the segmentation performance on linear structures.

2.1.3 Performance Evaluation

To evaluate a semi-automatic system, we need to ask the question whether the system is useful. The term *usefulness* was defined by Nielsen, as incorporating utility and usability [104]. Utility evaluates how well the system fulfills the functionality required for predefined tasks, and it is mainly presented with both quantitative and qualitative attributes. Usability, on the other hand, is a quality attribute that assesses how easy the user interface is to use.

Most reported evaluation criteria in semi-automatic image interpretation rely on utility. Researchers are concerned more on how well the developed models can solve the tasks. Some popular criteria include completeness, correctness, accuracy, and efficiency.

- **Completeness**

  Completeness is defined as the percentage of the correctly interpreted targets among all targets. For example, in building detection, it refers to the percentage of correctly detected buildings among all buildings in an image. This criterion has been used in [63, 140, 52].

- **Correctness**

  Correctness is defined as the percentage of correctly interpreted targets among all interpreted targets. **All interpreted targets may include both correct and wrong interpretations.** It is used in [44, 22, 3, 74, 151, 106, 105].

- **Accuracy**

  Accuracy is the degree to which a given quantity is correct and free from error. For example, in road detection, the accuracy of the detection can be evaluated by the average distance between the extracted and the true roads [52, 146, 147].

- **Efficiency**

  The efficiency is a time-related criterion. It is affected by two factors, the efficiency of the computational algorithm, and the time used for interaction. The latter can be considered partly as usability issue. Most image interpretation systems reported the efficiency in terms of average computation time for each task [99, 2]. Interaction time is normally evaluated by comparing the absolute time between the system and manual work, or by calculating the percentage of human effort saved [9, 50, 151, 106, 48].

- **Qualitative**

  Except for the quantitative evaluations above, many researchers demonstrated the quality of their models by showing images on the interpretation results. It is effective when quantitative evaluation is difficult, such as in evaluating a sequence of frames in a video [84], or is not necessary, such as in highlighting the interactive procedure [42, 37, 29, 36].

Completeness, correctness, accuracy and efficiency are quantitative criteria. They are suitable for testing and comparing different image interpretation methods and to identify their strength and weakness. One problem in using these criteria is that there is normally no standard image database available for testing purposes [1]. Thus, the comparison of evaluation results is not very meaningful. This is the reason why many researchers tend to use qualitative criteria, which can give a direct visual impression on the performance of the system. In the last column of Table 2.1, we give a summary on the criteria used for some semi-automatic image interpretation systems. We find that both qualitative and quantitative

[1] We need to point out that some researchers have paid attention to this problem. Mayer and Baltsavias have started an European Spatial Data Research which provides high resolution satellite images for road extraction study. Researchers can download images, run their own algorithms, and upload results for comparison from the following website: http://www.bauv.unibw-muenchen.de/institute/inst10/eurosdr/.

evaluation criteria are widely used. Another problem in the utility evaluation lies in the efficiency criterion, in which assessing the human involvement should be further considered besides the time for interactions.

From a software engineering point of view, usability is the most important factor for the success of a system. Nielsen pointed out that the usability should be evaluated using the following five quality components [104]. First, learnability, which refers to how easy a user can learn to perform a task the first time they use the system. Second, efficiency of use, which refers to how quickly a user can perform a task. Third, memorability, which refers to when user is away from the system for a period of time, whether or not they can get re-familiar with the system quickly. Fourth, few and noncatastrophic errors, which include the number of errors a user make, the Severeness of the errors, and whether or not it is easy for a user to recover from the errors. Finally, subjective satisfaction, indicating whether or not a user is having a pleasant experience using the system.

Among these five components, learnability, efficiency and errors should be paid more attention to in designing semi-automatic image interpretation systems. The speed of users performing a task and the time they spend to identify and correct errors greatly affect the overall efficiency of the system. Harvey reported their work in evaluating ROADMAP, which is a semi-automatic road extraction system [50], and they observed 20% or more improvements after users were familiar with the ROADMAP system. Even so, when comparing the system against completely manual work, the ROADMAP system was 12% slower than the manual work. The major problem was that users found the interface hard to control and found it difficult to switch between automatic and manual operation. This work suggests that, in evaluating a system, we need to consider both the computational factors and the user factors. In table 2.1, we give a summary on some typical semi-automatic image interpretation systems in their applications, interfaces, input data usage, and utility evaluation.

2.2 Related Work in Image Interpretation for Map Revision

In this section, we review related work in both automatic and semi-automatic image interpretation methods for map revision. We focus on the two most dynamic map features, roads and buildings.

2.2.1 Road Recognition

Road revision is one of the main tasks in map revision because roads are the most frequently changed map elements. In remote sensing images, roads are typically long, smooth, and continuous. In the following sections, we discuss several approaches in road recognition.

Knowledge-based Road Recognition

Knowledge-based image understanding techniques have been widely used in road recognition systems [24]. Knowledge refers to all information related to roads, such as their context in the image, intermediate processing results, defined rules and constraints, and so on. It also includes input data to be used in the processing [6]. The source of knowledge could be images, maps, road databases, regulations, human and so on. Some reported knowledge-based road recognition systems are [12, 72, 91, 120, 126, 135, 144], sone of which are discussed below.

Mayer and Steger studied scale-space events to extract road features [91]. The use of multiple scales is based on the knowledge that objects display different properties according to the scale of the image. High-frequency information may be removed at a coarse scale, and thus facilitate the extraction of object features. Multiple-scale images can be derived by Gaussian filters or by mathematical morphology. The authors extracted 1D bar-shaped profiles with width and height information, as well as 2D curvatures based on the 1D profiles at multiple-scales. To link the scale-space events, the width of roads and with of the gap between roads were analyzed and an appropriate scale was selected. This assured that unwanted substructures, like cars, were eliminated at a symbolic processing level. Finally, the road features, from fine to coarse scale, were integrated to generate a highway network using a knowledge-based model.

Katartzis *et al.* proposed an automatic road extraction model from airborne images based on local and global analysis [72]. In the local analysis step, human knowledge on roads was applied to build geometric and radiometric models. A morphological approach was used to detect elongated structure with a certain width based. Then a line following algorithm was used to detect line segments that were longer than a predefined threshold. In the global analysis step, the detected line segments were grouped into roads using a Bayesian method. First, a graph was constructed to characterize the connectivity among line segments. Then road identification was carried out using Markov random field based on Bayes and MAP criteria to maximize the posterior probability of the labelling given the graph model. Several aspects were investigated that would influence the grouping, such as

the parameters in the graph creation, the formulation of probability distributions, and the reduction of the number of parameters.

Bentabet *et al.* implemented a road vector update system using two sources of knowledge: SAR imagery and road databases [12]. The road database was used as a reference for estimating the initial road location given that the database coordinates are normally close to the true roads location. Then roads were detected from the SAR imagery using snake approach and were used to update the road database. The road detection system was composed of three modules, preprocessing, line extraction, and road detection. The preprocessing module used a Frost filter to remove unexpected lines from the image and to smooth the roads. Then a line detector based on Canny's criteria was used to extract line feature. These were used as input to the snake in calculating the external energy. This work provides a good example on how knowledge can be used to compute the parameters of vision algorithms.

Shackelford and Davis processed high-resolution multispectral images using a pixel-based and object-based approach [120]. The pixel-based approach classified image pixels into four classes, Grass-Tree, Road-Building, Water-Shadow, and Bare Soil, using spectral and spatial knowledge. This image was further segmented using region-growing. Then the object-based, fuzzy logic classifier was called to label each segment. In this step, shape, neighborhood information and spectral statistics were used as object features. A correct classification rate of 86.4% was achieved.

A summary of knowledge-based road recognition systems is given in Table 2.2. In these systems, knowledge from different sources have been successfully used for different application purposes. Common usage of the knowledge is that it guides the design and development of computational models. As we will see in later chapters, such knowledge is static, and thus cannot characterize the dynamics of image features. To make systems more robust, additional human knowledge should be incorporated into the image interpretation process through human-computer interactions.

Road extraction and tracking

When classified by purpose, the processing of roads is often divided into two main processes, road extraction and tracking [150]. In road extraction, typical sequences of road points are detected within an area of interest. Such road extraction systems include [150, 92, 76, 127, 69]. In road tracking, a part of the road is already known and different strategies are used to track the road [93, 40, 11, 8]. These systems are discussed in the following.

Table 2.2: Summary of knowledge-based road recognition systems

systems	purpose of systems	knowledge sources
Bentabet [12]	road update	image and road database
Katartzis [72]	road extraction	human
Mayer[91]	road extraction	human
Shackelford [120]	scene classification	spectral and spatial property
Ton [126]	image segmentation	spectral and spatial property
Wang [135]	highway network extraction	reasoning rules
Zhang [144]	road update	road database and road design rules

Klang was one of the first researchers to use snakes in road extraction [76]. Similar to [12], he also used a road database and satellite images as source of road database updating. Road end points and intersections were extracted from the road database in order to get node points. These node points were matched to the neighborhood image pixels using iterative least square matching. Connecting the nodes in the image formed the initial road segments. A ziplock snake was then used in road delineation. This approach was used in mapping rural areas. Similar work including the road system based on scale-space and snakes was reported by Laptev *et al.* [80].

Tupin *et al.* introduced a road extraction method on urban areas using multiple-view images [127]. These images were taken from two flight directions that were perpendicular to each other. Like many other systems, this method was also composed of line detection and road network construction steps. But then a merging method was applied, in which the road networks extracted from the two views were superimposed. The resulting road detection rate were increased by about 30%, compared to the single view detection rate. During the road network reconstruction, Markov random fields (MRF) were used to label the roads from a set of candidate road segments. A similar system, also using of MRF was reported in [72].

Hinz and Baumgartner interpreted multiple views in another way. They correspond views to the resolution of the same image [58]. Different parts of the road model and extraction strategy were performed differently in these views, and thus an optimal exploitation was achieved. Another work using multi-resolution image processing was reported in [53].

Some researchers used 3D topographic information in road extraction. The 3D information can help to distinguish buildings and trees from roads. Zhu *et al.* proposed a system to detect hidden road edges covered by shadows of high objects in urban area [149]. After

high objects were removed, the road centerline was reconstructed by spline estimation.

Jeon *et al.* modelled roads in a spaceborne SAR image as curvilinear structures with width information [69]. The curve segments were grouped using a genetic algorithm. The grouping started from an initial seed segment selection among the curve segments. Region growing was incorporated into the genetic algorithm so that only segments within the neighborhood of the seed were input to the algorithm. Then the grouping was performed by the genetic algorithm using each seed. This model detected roads with an accuracy of 92.2%, and with average error of 2.08m.

Geman and Jadynak provided a partial solution for tracking road networks from medium resolution satellite imagery [40]. Given a starting point and direction, the tracker performed on-line testing of road hypotheses using decision trees. During the testing, a geometric model and a statistical model were used to constrain the search direction, and a local filter was designed to extract possible road segments. The on-line testing adopted an entropy testing rule, and generated an approximation to the maximum likelihood estimation. This solution successfully extracted highways over one hundred kilometers without human involvement.

Kim *et al.* proposed another system for road centerline tracking [73]. The tracking started from user defined road templates, which were used to match the image along the direction of the road using least squares correlation method.

Table 2.3 shows a summary on typical work in road extraction and tracking. Comparison are made on road operations, type of images used for the analysis and experiments, as well as major methods used for image processing, feature extraction, road detection and classification. We do not compare the performance of each system because there is no standard database available. This summary is far from complete since there are numerous research papers published in these topics.

The summary shows that many image processing and machine learning methods have been successfully used for the road extraction and tracking. Main methods for image processing include image segmentation and morphology. Edge and curve detection are the key methods for feature extraction. In the classification step, methods include decision tree, support vector machine, Bayesian modelling, and others. These facts leave us many choices when semi-automatic road recognition algorithms are developed. We have to look into each step and explore the optimal combination of these methods, as well as their incorporation into the interactions.

Table 2.3: Summary of road extraction and tracking.

systems	road operations	type of images	methods
Heipke [53]	extraction	aerial	multi-resolution processing, data fusion
Hinz [58]	extraction	aerial	multi-resolution processing, data fusion
Jeon [69]	extraction	satellite	curve detection, Genetic algorithm
Katartzis [72]	extraction	aerial	morphology, MRF
Klang [76]	extraction	satellite	snake
Laptev [80]	extraction	aerial	multi-scale processing, snake
McKeown [92]	extraction	aerial	edge analysis
Priestnall [112]	extraction	satellite	Bayesian modelling
Song [122]	extraction	satellite	image segmentation, SVMs
Tupin [127]	extraction	satellite	MRF, data fusion
Zhu [149]	extraction	aerial	edge detection, spline approximation
Zlotnick [150]	extraction	aerial	edge detection
Barzohar [8]	tracking	aerial	stochastic modelling and estimation
Geman [40]	tracking	satellite	model-based tracking, decision tree
McKeown [93]	tracking	aerial	edge detection
Kim [73]	tracking	satellite	least squares correlation matching

Automatic versus semi-automatic methods

Road processing can also be classified as automatic or semi-automatic depending on whether the human is involved in the process. Automatic methods aim at replacing humans [40, 76, 58, 92]. However, humans cannot be excluded completely from the map revision process in real applications because the image interpretation algorithms are not sufficiently robust and reliable and, importantly, a map is a legal document, requiring some form of final checking by a human. For these reasons, semi-automatic methods are preferable [80].

KcKeown and Denlinger [93] introduced a semi-automatic road tracker based on road profile correlation and road edge following. The tracker was initialized by a human operator to obtain starting values for coordinates, direction and width of a road. Road axis points were then predicted by a road trajectory model and correlation models. The edge-based tracker modelled the road by linking points with high gradient and orientation in the expected direction.

Vosselman and Knecht [134] proposed a road tracker based on a single observation model Kalman filter. Human input was used to initialize the state of the Kalman filter and to extract a template road profile. The Kalman filter then recursively updated its state to predict the road center, using feedback from matching the template profiles to the observed profiles. Baumgartner [9] developed a prototype system based on the above method. An interaction interface was designed to coordinate human actions with computer predictions.

Xiong and Sperling presented a semi-automatic tool that enables human to visually check and correct mistakes in an automated algorithm for performing road network matching [140]. Similar to [69], the automated algorithm started from road seed selection. These seeds were extracted from latest aerial images. A clustering procedure was used to identify matches between the seeds and the existing database. A matching table and a matching map were generated, and these results were checked and verified by human operator. The matching procedure then went through segment mapping and edge mapping steps. The effectiveness of the tool was evaluated by the matched road length.

More recent semi-automatic approaches include the least squares template matching methods [46] for road centerline extraction by [63] and [73], both requiring seed-point input from humans to generate 2D template.

Harvey *et al.* looked at the semi-automatic road extraction from a different point of view [50]. They performed a usability evaluation on the ROADMAP road network tracking system against pure manual systems. Users were required to perform automatic, semi-automatic, and manual road tracking using two software packages, and statistic analysis was made to the recorded time costs. The authors concluded that the ROADMAP was comparable to commercial software.

2.2.2 Building Extraction

In this section, we review some of the latest work on building extraction from aerial images. This has been an active topic for almost twenty years because of its application in computer aided cartography, remote sensing and geographical information systems. These systems are either automatic or semi-automatic. A more complete survey of systems before 1999 can be found in Mayer's paper [90]. This paper introduced and assessed several examples of building extraction approaches in terms of complexity of data, characterization of models and characterization of strategies.

Cord, Jordan and Cocquerez developed a hierarchical system to automatically extract and model buildings from stereoscopic pairs of high resolution aerial images of urban areas [23]. The image pairs were input into the system and processed using a stereo matching scheme to compute a dense Digital Elevation Model (DEM). A matching cost function based on gradient vector correlation for edge-adaptive windows was developed in the matching scheme. A global scene classification based on altimetric data was then performed based on the DEM. The scene was classified into ground regions, building regions and vegetation regions. For each building detected, two-step individual building modelling

was performed. The first step was to deal with the 2D roof boundary modelling using the 3D and radiometric information. The second step was to model the roof surface based on a parametric multiplane 3D data model and a stochastic algorithm.

Fradkin, Maitre and Roux introduced another automatic building detection system from multiple aerial images in dense urban areas [35]. The approach consisted of two steps, surface reconstruction and building detection. The accuracy of the first step decided the reliability of the second step. To achieve accurate surface reconstruction, a multiview stereo matching followed by robust surface interpolation was implemented. Dense elevation maps for the most oblique views of the scene were constructed into the first step. Then several oblique views were used to detect the ground projection of building facades in object space. Building hypotheses were established using the facade information and prominent planar 3D regions from surface reconstruction. Finally, these hypotheses were verified using a set of conformation criteria. A 80% good detection rate and high reliability (near 99%) was reported.

Gulch reported a semi-automatic building extraction system, which provides a tool to derive reliable 3D geometric information on buildings [47]. The inputs were two images, image orientation and the average terrain height. To find the homologous points from two images for 3D construction, the human selected a small area in the first image, and the computer automatically selected the homologous point in the second image by searching for the maximal normalized correlation coefficient. The operator could modify the length of the roof and rotation with mouse. The roof-top height matching and form adaptation were done automatically by the computer. Finally, the ground height was determined by selecting one point in one image and an automatically determined corresponding point in the other image. The system produced 3D models of buildings in vector format for which only four mouse clicks were required.

Caelli, McCabe and Briscoe developed a shape tracking and production system to generate shape boundaries from image features using Hidden Markov Models [17]. The shape boundaries were defined by sequence of states with each a probability distribution of expected image feature types. In the boundary tracking and generation process, the initial model estimates were obtained from observed expert behavior and then updated by the Baum-Welch estimation procedure. Following this, a new version of the Viterbi procedure was used to determine the degree to which each HMM predicted shape states sequences from observed sequence of feature types. Hamming distance between predicted and observed symbol sequences using a MonteCarlo method was used to evaluate the model per-

formance.

Huguet, Carceroni and Araujo described a solution to the problem of recovering the 3D structure of an urban area from a set of low-altitude aerial images [65]. To constrain and guide the 3D reconstruction, prior knowledge on a large set of architecture scenes were used instead of an a priori geometric model of a single building. Homogeneous features, such as roofs, walls and pavements of the objects, were used to segment these objects in images, using a proposed Colored Watershed method. To build the 3D reconstruction after the initial segmentation, a dense elevation map was produced using dense-stereo algorithms. The proposed algorithm performed the shape estimates through the use of segmentation and stereo techniques from the raw images and buildings. These shape estimates were refined in a final, image-driven optimization step.

Significant research has been done on automated cartography by the Digital Mapping Laboratory in School of Computer Science at Carnegie Mellon University, including automated phototexturing from aerial and terrestrial imagery, feature extraction, road extraction, stereo processing and generalization [92]. For example, the automated phototexturing from aerial and terrestrial imagery started with the building of a phototextured model of the CMU campus. The major steps included resecting aerial and terrestrial imagery, building extraction, terrain skin generation and texture selection methods. The building extraction used a semi-automated multi-image site modelling tool called SiteCity, which was based on the GOMS (Goals, Operators, Methods and Selection Rules) HCI principles [62]. Image understanding algorithms were integrated into SITECITY to assist human operators. The evaluation showed that semi-automated processes in SITECITY enhanced the overall usability of the system and reduced the complexity of manual mensuration of three dimensional building objects.

Shufelt addressed the evaluation of unbiased metrics for qualifying detection and delineation performance, applying these metrics to a representative body of test images, and understanding the impact of image and scene content on building extraction systems [121]. He evaluated four existing monocular building extraction systems, using image-space and object-space based metrics on test images from different sites. The effects of image obliquity and object complexity on system performance, and the effects of edge fragmentation, were also analyzed. Even given such efforts, there is to date not a single system that can reliably perform building extraction.

2.3 Summary

In this chapter, we have reviewed related work in semi-automatic image interpretation and its application to map revision. The review focused on two major topics, the human-computer interaction for image interpretation, and the semi-automatic road recognition and building detection. From the review, we can draw several conclusions. First, the semi-automatic method is a practical way to address the lack of robustness in image interpretation. Second, research in human-computer interactions for image interpretation is still limited, especially in modelling the user action and developing interactions. The former answers the question what we can do, and the later tells us how we can do. Third, current research achievements in automatic methods provide us with many choices when a semi-automatic method is exploited. These choices form the basis of our study and what we need to do to combine selected methods with human-computer interactions.

Chapter 3

HCI Framework for Image Interpretation

3.1 The Framework

The aim of this thesis is to build an HCI framework for cartographic map revision, which uses image interpretation methods as the revision approach. We first discuss a general HCI framework for image interpretation systems and then apply it to semi-automatic map revision.

Most image interpretation research is aimed at replacing humans in performing perceptual tasks. We noticed that very few automatic or semi-automatic approaches have been used in map revision or remote sensing systems. Although some GIS system provides tools for automating tasks to a certain degree, these tools are restricted to post editing of acquired data. Classic semi-automatic systems developed in universities or labs allow a human to initiate the image interpretation tasks by inputting initial features, but then proceed with little or no HCI [64, 95, 108, 93, 9, 25]. Unfortunately the accuracy, robustness and reliability of such approaches are far behind real-world requirements. What is required is more dynamic on-line interactions between image interpretation routines and human operations. Such systems demand integration of several components, namely a human-computer interface, user modelling, image interpretation models, knowledge transfer schemes, and performance evaluation. As Barnard *et al.* pointed out [7], developing a general HCI theory is difficult because there are too many types of users, computers and applications. The same applies to the development of a general HCI framework for image interpretation systems. However, we believe that any successful system should at least consist of all the five components above, interacting in the way shown in Figure 3.1.

Within this framework, human operators generate inputs through the human computer

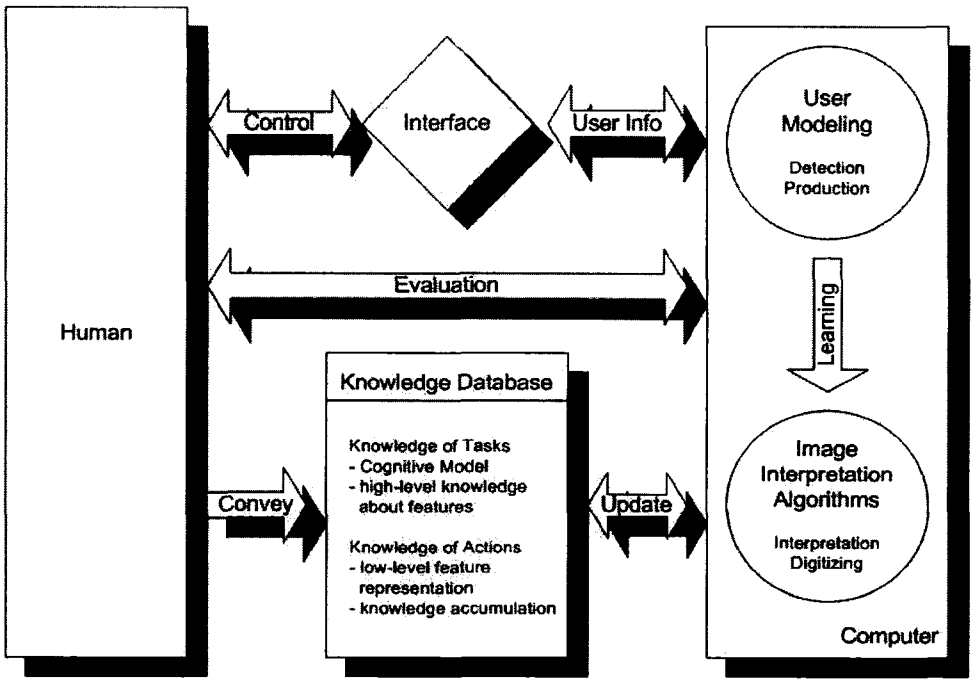


Figure 3.1: An HCI framework for image interpretation systems.

interface. The inputs can be used for multiple purposes as summarized in Section 2.1.2. Before any automatic image interpretation methods are developed, analysis of complete manual image interpretation can help understanding the difficulties in the interpretation tasks. This requires modelling of the human behavior in the system. Besides computational analysis, user modelling can also be performed by heuristic usability analysis, including questionnaires, user surveys, and direct monitoring of human behavior. With the acquired user model, we can replace human with image interpretation models in simple tasks. The direct development of a robust model is difficult, even though offline training may be available. An advantage of this human-computer interaction framework is that the update of image interpretation models can be performed under the guidance of a human. Human knowledge can be transferred to the computer through a knowledge database or through online machine learning. The control of the system and the knowledge transfer is achieved by the performance evaluation and through the human-computer interface.

3.2 Human-computer Interface

The choice of the interface has a significant effect on what can be attained. When a keyboard is used, human input can be given as parameters or labels, and thus can affect the image interpretation models directly. When a mouse and other input devices are used, the

inputs should be analyzed and transformed into parameters for automatic models. In more complex input interfaces, such as gestures and speech recognition, systems must be developed to interpret the user's intention.

Within the user interface, two tasks should be performed. First, the human inputs should be tracked and recorded. This is the syntactic level of input analysis. Second, the recorded inputs should be interpreted into actions, so that the computer can understand what the user is doing. This is the semantic level of input analysis. Both tasks are non-trivial. Due to a lack of understanding of the software mechanism and unavailability of the source codes and packages, it may be hard to record user actions. Although researchers can work with small applications or try to build simulation systems by developing simple interfaces for human computer interaction, the user action data collected in these simplified human-in-the-loop systems is very restricted compared to the complex human behavior in an unconstrained real world. Even when the actions can be acquired, further understanding of human intentions requires complex analysis and modelling.

Another issue to be considered is the way users interact with the system. An interface that is difficult to learn and to use can deeply affect user performance, and in turn, can affect the overall performance of the semi-automatic system. Thus, usability evaluation should be considered in developing an interface.

3.3 User Modelling

Modelling user behavior provides quantitative information for computers to learn about human actions and how specific computational image interpretation procedures can replace humans [115].

By monitoring the process in human image interpretation (detection) and annotating the interpretation results (production), we can get statistical data on the types of tasks that take a human more effort to finish. Then we can study whether and how these tasks can be automated. Such monitoring can be performed by direct observation of human behavior by an expert or by analyzing recorded human inputs in a software system.

User modelling can also help to determine how humans perform the tasks and to predict user actions. We need to find out the interconnections between image features and human actions. We also need to analyze the relationship between individual actions in an action sequence. Such modelling helps the development of robust computational algorithms and user friendly systems.

3.4 Image Interpretation Models

Image interpretation models perform the perceptual tasks to replace human. Normally, complete image interpretation models include algorithms for preprocessing, feature extraction, object recognition and classification.

The purpose of developing an interactive system is not do explore new image interpretation models. Instead, we have to find out how the existing models can fit into the interaction process. Choices need to be made to select from a number of current models for the design of a system. On the one hand, good computational models are not necessarily suitable for human-computer interactions because they may not require human involvement or may not suitable for the designed interactions. On the other hand, some models may not be robust at the beginning, but can evolve progressively with human guidance.

3.5 Knowledge Transfer Schemes

In the human-to-computer knowledge transfer, both static and dynamic knowledge is relevant. Static knowledge reflects conceptual information about specific tasks. It typically defines the types of image interpretation algorithms to be used. It also guides the selection and extraction of features, such as color, shape, texture, color layout and segmentation [117]. Dynamic knowledge reflects the diversity of the real world. When applied to image interpretation applications, it indicates feature changes, the main reason why most automatic systems fail. Dynamic knowledge can be transferred to the computer through knowledge database construction and online learning.

A knowledge database contains information on the specific tasks that are to be performed. The information includes the cognitive model how humans perceive an image feature and general knowledge on how the feature should be used. For example, in an aerial image, roads should be long, smooth, continuous, and homogenous. This knowledge forms the basis of the prior probabilities used in the machine learning routines of each task. These priors are recursively updated by the results of the image interpretation algorithms and the actions of the human operator. These updates form a lower level of knowledge representation in a knowledge database, including low-level image feature representation and knowledge accumulation.

Combining user modelling, image interpretation models and the knowledge transfer schemes, it is possible to develop user adapted systems. Specifically, we can decide what functions are to be served by the adaptation, what properties of the user should be modelled,

what methods should be employed to infer user goals, and how the system should behave adaptively, and so on [68].

3.6 Performance Evaluation

Performance evaluation has two levels, the system level and the internal level. At the system level, the evaluation can be made according to the criteria introduced in 2.1.3. At the internal level, the performance evaluation is a two way process through HCI, the computer side and human side. On the computer side, the evaluation of human inputs enables the image interpretation system to eliminate noise from human inputs and decide whether to accept or reject the inputs. The computer also evaluates its own performance and allows the human-in-the-loop to gain control over the whole process. On the human side, when control has been given up by the computer, the human evaluates the situation and decides the next step the system should take.

Chapter 4

Applying the HCI Framework to Map Revision

Map revision is traditionally a manual task, especially when maps are updated on the basis of aerial images and existing map data. This is a time consuming and expensive task. For this reason, maps are typically out of date. For example, it has been reported that, for a number of reasons, the revision lag-time for topographic maps from the United States Geological Survey (USGS) is over 23 years [45].

We applied our HCI framework to the application of map revision. In this chapter, we first review the current status of topographic map revision. Then we report the interface analysis in the software environment for the USGS map revision program.

4.1 Status of Topographic Map Revision

The USGS is the major federal agency in the United States for the collection and distribution of digital cartographic data. A similar role is played by the Earth Science Sector of Natural Resources Canada (NRCan).

4.1.1 USGS Map Revision Program

The primary product of the USGS National Mapping Program is the 1:24,000, 7.5-minute topographic quadrangle series, which is the only uniform map series that covers the entire area of the continental United States in considerable detail [96]. This map series includes about 53,000 map sheets, the building of which lasted from the mid-1940's to the early 1990's.

The revision of this map series started from the beginning of the program. From the mid-1960's, the revision number became significant. To keep the primary series map cur-

rent, the USGS began a revision plan in 1992 based on user requirements, available resources and funding. For example, maps that are demanded most are given priority in the revision work.

To speed up the revision process, stereophotos without field verification are used to collect new features. However, the revision speed is lagging far behind the requirements. In 2000, the median map date for the series was 1979, which means that their median age was over 20 years. In order to keep the current age of maps, at least 1,500 maps should be revised every year. Furthermore, the aerial photos and other source materials used for map revision are usually 3 to 5 years old, which makes even the latest maps out of date.

Four types of revision are performed in USGS, minor revisions, basic revisions, complete revisions and single-edition revisions. Minor revisions use aerial photos to update the few changes since the last revision. Basic revisions use Digital Orthophoto Quadrangle (DOQ) and aerial photos to update map features and costs about USD17,000 per quadrangle. Complete revisions and contour updates are rarely done because of high costs.

4.1.2 Digital Orthophoto Quadrangle

The data sources for map revision are original maps, recent aerial photos and DOQs, as well as information from other sources such as other government agencies. Among them, the DOQ is the most critical input because it is positionally more accurate than all other sources. Features such as roads and buildings can be directly collected from DOQs. An objective of basic revision is to make the revised map match the DOQ.

DOQs are produced from aerial photos taken at a height of 20,000 feet with an approximate scale of 1:40,000. When digital orthophotos are produced, several inputs are required besides the original perspective image, namely digital elevation models, ground coordinates of ground control points, camera calibration information and the user parameter file. DOQs can distinguish ground objects of 1 meter size for quarter-quad digital orthophoto and of 2 meter size for quadrangle digital photos, which is sufficient for detecting roads and buildings.

During the process of map revision, the map of a quadrangle is scanned into a computer at 1000dpi and becomes a raster image. This raster image is displayed over the corresponding DOQ on the computer screen and uses the DOQ as the source of revision. The cartographers then make a visual comparison of the raster image and DOQ. Since the scanned features appear on top of the DOQ, feature changes such as shape, size and distance can be easily detected.

4.1.3 Map Revision in Canada

The center for Topographic Information of Natural Resources Canada produces National Topographic Systems (NTS). The NTS series provides general-purpose topographic coverage of Canada at the 1:50,000 and 1:250,000 scales. NTS maps depict terrain features (landforms and land cover), drainage (lakes, rivers and streams), official boundaries, the transportation infrastructure (rail and road networks) and many other man-made features (buildings, power lines, pipelines, dams, cut lines, *et al.*).

Very little revision of NTS map sheets is currently being done. In many cases, the revision process is partial, in which only features such as the road network, boundaries, and toponymy are revised, along with the addition of metric contours derived from Digital Elevation Models. In other cases, a more thorough revision is done using satellite imagery.

The NRCan's Earth Science Sector is gradually abandoning the revision job of NTS because of shortage of budget and personnel. In turn, they began to work on a project of GeoBase, which is an initiative of the Canadian Geospatial Data Infrastructure. This database will become the source for new map products.

The GeoAccess Division of the Geomatics Sector of NRCan is another provider of the geographic data in Canada [49]. Its main product is the National Atlas of Canada, which is a multi-themed small-scale map (1:6,000,000). During its revision in 1997, a large amount of manual compilation was done because much new information from multiple resources was not in database.

4.1.4 Other Map Revision Programs

Most countries have map revision programs although many suffer from the same finding and logistic problems identified above.

In the United Kingdom, Ordnance Survey (OS) is the leading agency in providing geographic data. One of the major products, the OS MasterMap, provides topographic information on nine themes representing layers such as land area classification, buildings, roads, water, terrain and height, boundaries, and others. This product is created from a master database that is updated every six weeks. The updating uses the old database and the latest aerial photos as the data source, which requires cartographers to make visual comparison of the two, digitize the changes, and to modify the database.

In Finland, the topographic database contains nine data groups, such as transportation network, transmission network, terrain, hydrography, elevation and buildings [54]. The revision of topographic maps uses 1:16,000 or 1:31,000 aerial photos for level A building,

power lines, fields, water bodies and roads. The aerial photos are scanned using 20 micron resolution. Aerial triangulation, digital elevation model editing are then performed. Operators use photo interpretation to determine the changes that happened in the field, new objects are added and associated to existing objects, and disappeared objects are deleted.

In China, the State Bureau of Surveying and Mapping (SBSM) finished building the National Fundamental Geographic Information System in February 2006. This system include 1:50,000 topographic maps that cover mainland China, Hongkong, Macau, and Taiwan. The update of this database started in June 2006, and will last for 5 years. 19,000 topographic maps will be updated using 1:10,000 map and remote sensing images as the source of revision.

4.2 Human-Computer Interface for Semi-automatic Map Revision

We use the USGS topographic map revision system as the general platform for our research. Current USGS maps are printed on white paper with six colors: black, red, brown, green, blue and purple, one for each feature. The revision of this map series is the Raster Graph Revision (RGR) program. 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 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. 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 4.1 shows the environment of the RGR system.

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.

To apply the HCI framework to map revision tasks, we have explored a prototype that consists of following components: an interface to track and parse human actions in map revision, a model for user behavior patterns to support and automate the map revision process, image interpretation algorithms to replace simple subtasks performed by humans such as

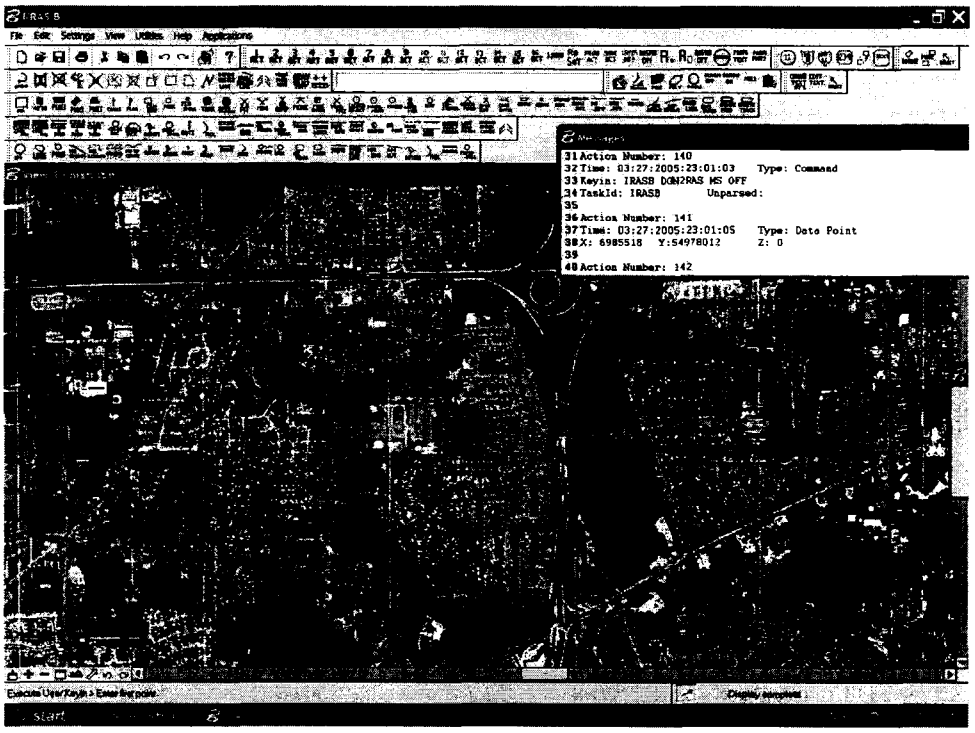


Figure 4.1: Map revision environment in the USGS. Here previous map layers are aligned with current digital image data. The icons in the tool bar correspond to operations on feature collection and environment setting. The pop-up windows on the upper-right screen is an interface to record system-level events.

road section tracking, processes whereby the computer selects and tunes image interpretation algorithms from its ability to predict human behavior throughout the tracking process, and evaluation criteria for purposes of feedback so that the computer can quantitatively evaluate human inputs and its own performance. Each of these components is discussed in detail in the following sections.

4.2.1 Tracking User Actions

In the Bentley Microstation, users interact with the system using keyboard or mouse. A simple drawing operation may be implemented by either clicking a tool icon on the tool bar or by entering a key-in command. To facilitate map revision, RGR implemented a set of tools for cartographic symbols, each of which encompasses a sequence of key-ins. At the system level, the mouse input is also transformed into key-ins. Each key-in is considered an event. Events from both inside or outside Microstation are processed by an input handler and are sent to a queue, and a task ID is assigned to each event.

With the imbedded Microstation Development Language (MDL) environment, we can

keep track of the states of the event queue and extract detailed information of each event, for example, the task ID, the key-ins, the time, and the coordinate of the mouse clicking, and so on. To fully describe an event, we used the following data structure:

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

Table 4.1: Data structure for system-level event

Doing so, we have fully captured and recorded the time-stamped system-level event sequence. This sequence contains both inter-action and intra-action information. Then we have to parse the sequence into a meaningful higher-level user action sequence. This step is similar to natural language processing.

4.2.2 Analyze and Parse User's Actions

Altogether there are 278 tools in RGR software, each corresponding to a user action. This number could be increased when new standard are utilized in USGS. An analysis of all these tools is not necessary. First, some tools are used for features that rarely appear or that need not to be revised in most cases. Second, some tools relate to the registration of the scans and DOQs, environment setting, file input/output, and map plotting. Such task are not involved in the feature collection process. Thus, we have made the following assumptions:

- We are only interested in the actions related to feature collection.
- All the environment settings have been done before the tracking of user actions.
- We are performing an offline analysis.

With these assumptions, the number of tools is reduced to 144. Each of these tools is composed of a fixed sequence of events.

When doing the revision, a complete action may contains a tool selection and an intervening sequence of coordinate clicks, view changes, as well as one or more reset operations which are used to end the coordinate clicks and the action. For each action, the corresponding sequence of events involved in the tool selection is fixed. View changes may happen

both before and after the coordinate selections. The reset patterns are different across the class of actions. In analogy to natural language, a complete action can be viewed as a sentence. The tool selection, coordinates clicks, view changes, and reset can be viewed as words. Thus, syntactic and semantic information is contained both at the sentence and word levels. Depending on the how the specific feature looks and the way that user perform the action, the sequence of the sentences and words may be different in each revision task. It makes the parsing process a challenging task.

Lexicon

We built a semantic lexicon to store the word information. The lexicon has two parts. The first part contains the spelling information of each word. Because each word correspond to an identical action, the spelling sequence is the same as the sequence of events in tool selection of the action. Table 4.2 shows the spelling information for drawing class 1 road, which is a list of events. The spelling information uses the same data structure described in Table 4.1. "Type" indicates the type of the input, namely the keyin, coordinates, or reset. "Keyin" is the actual key-in command. "TaskID" indicates the event source, *e.g.*, where the event comes from. "Unparsed" indicates the unparsed component of the key-in that needs to be processed by the system. Note that the spelling information does not contain the event ID, the event time, and the coordinates, which can only be captured in the drawing process.

The second part is a semantic marker which shows the usage of the action. It contains the information like how many coordinate clicks and resets should an action have and the meaning of the position of the coordinates in an action given the resets.

Depending on the action usage pattern, we classified the actions into 17 groups. Table 4.3 shows the semantic part of the lexicon.

Parser

A parser was designed according to the syntactic and semantic information of each identical action. A scan generator is first applied to the sequence of system-level events. The events are grouped into words according to the spelling information of each action. Then, the sequence of words is segmented into sequence of sentences, or complete actions, according to the syntax in Table 4.4.

The sequence of actions are arranged into a tree structure shown in Figure 4.2. The root of the tree is a project, which is defined as the revision of a map. The task is defined as the revision of one ground object, such as a road, a block of buildings, a lake, and so on.

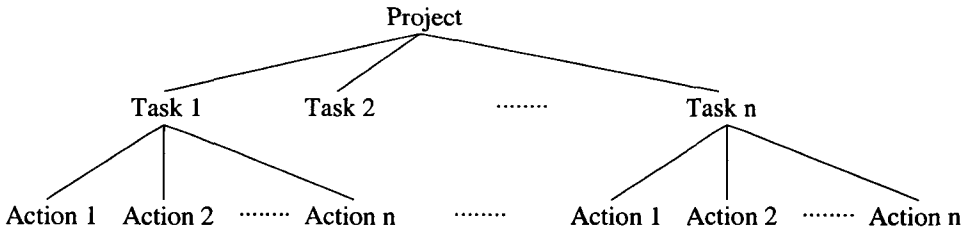


Figure 4.2: Hierarchy of the action database

The project- and task-levels contain the semantic information of the map revision process and can be tagged by human input. Finally, the user actions are stored into an XML format database. Table 4.5 shows an example of the human input during road tracking, which is extracted from the XML database.

4.3 Conclusion

This chapter has reviewed the current map revision programs, one of which, the USGS map revision system, has been introduced in detail as the software environment that we are studying. We developed a human-computer interface for tracking and recording human action as a time-stamped sequence of events at the system level. These events are parsed into an event sequence that is represented in XML format, and stored in a database. This is, as far as we know, the first open database on user behavior in real world applications involving document processing, feature tracking, or pattern recognition. It is the basis of our research in user modeling and semi-automatic image interpretation that are reported in the following chapters.

Type: Command TaskId: DRAFTPAL	Keyin: DRAFTPAL KEYIN macro 40'solid6 Unparsed: macro 40'solid6
Type: Command TaskId: USTN	Keyin: MACRO 40'solid6 Unparsed: 40'solid6
Type: Unknown	TaskId:
Type: Command TaskId: IRASB	Keyin: IRASB DGN2RAS MS OFF Unparsed:
Type: Unknown	TaskId: MBE1
Type: Unknown	TaskId: MACRO_40'SOLID
Type: Unknown	TaskId:
Type: Command TaskId: IRASB	Keyin: IRASB ACTIVE LAYER black12.rle Unparsed: black12.rle
Type: Command TaskId: IRASB	Keyin: IRASB IRASB refresh dialogs Unparsed: refresh dialogs
Type: Unknown	TaskId: MBE1
Type: Unknown	TaskId: MACRO_40'SOLID
Type: Unknown	TaskId:
Type: Command TaskId: USTN	Keyin: PLACE LSTRING POINT Unparsed:
Type: Unknown	TaskId: MBE1
Type: Unknown	TaskId: MACRO_40'SOLID
Type: Command TaskId: USTN	Keyin: MDL UNLOAD inputMon Unparsed: inputMon

Table 4.2: Spelling information for class 1 road

Group	Number of Coordinates	1st reset	2nd reset
G0	$n = 0$	N	N
G1	$n = 1$	Y	N
G1	$n = 2$	Y	N
G1	$n = 3$	Y	N
G4	$n > 1$	Y	N
G5	$n > 2$ and $n > 1$	R	Y
G6	$n > 2$	N	N
G7	$n > 2$ or $N = 1$	R	Y
G8	$n = 2$	R	Y
G9	$n = 1$ or $n = 2$	R	Y
G10	$n > 1$	R	Y
G11	$n > 2$	Y	N
V1	$n = 2$	Y	N
V2	$n = 1$ or $n = 1$	Y	N
V3	$n = 0$	N	N
R	$n = 0$	N/A	
CO	N/A	N/A	N/A

Table 4.3: Semantic lexicon. The first column is the group number. The second column is the number of coordinates contained in an action. The third column is whether a reset operation is required to end an action. 'Y' means yes, but can be omitted; 'N' means no; and 'R' mean required. The fourth column is whether a second reset operation is required. G1 to G11 are drawing action and drawing setting operations. V1 to V3 are viewing change operations. R is the reset operation itself. And CO is the coordinates of the action.

S	→	G0 | G1+S1 | G2+S2 | G3+S3 | G4+S4 | G5+S5 | G6+S6 | G7+S7 | G8+S8 | G9+S9 | G10+S10 | G11+S11 | R | CO | V
S1	→	I+F
S2	→	I+I
S3	→	S2+S1
S4	→	T+F
S5	→	S6+R+S2+F
S6	→	I+T
S7	→	S6+R | S1
S8	→	S2+S4
S9	→	S2+F | S1
S10	→	S2+M+R+F
S11	→	S6+F
T	→	S2+M
M	→	M+I | e
I	→	VI+CO
VI	→	VI+V | e
CO	→	CO | e
F	→	R | e
V	→	V1+CO+CO+F |V2+C+F | V3

Table 4.4: Grammar for parsing the system-level events

x	y	time
6982738	55013871	06:25:2005:12:25:37
6983315	55013274	06:25:2005:12:25:39
6983911	55012498	06:25:2005:12:25:41
6984388	55011862	06:25:2005:12:25:44
6984866	55011115	06:25:2005:12:25:45
6985174	55010514	06:25:2005:12:25:53
6985552	55009569	06:25:2005:12:25:55
6986148	55007798	06:25:2005:12:25:58
6986278	55007267	06:25:2005:12:26:05
6986397	55006341	06:25:2005:12:26:07
6986476	55005128	06:25:2005:12:26:11
6986516	55004839	06:25:2005:12:26:13

Table 4.5: Human input on road centerline during manual road tracking. It includes the x and y coordinates of the mouse clicks on the revision platform and the time of the clicks.

Chapter 5

User Modelling in Map Revision *

5.1 Introduction

The purpose of computer aided map revision is to improve both the speed and accuracy of the map revision, as well as to release humans from this tedious process. One solution to the tasks is to reduce human involvement in the feature collection and map drawing, by automatic feature tracking based on computational image interpretation. Until now, all research efforts in image interpretation fall in this area. However, the efficiency and accuracy of the image interpretation models are not necessarily consistent with human performance, suggesting that humans should be part of the feature tracking process. The human behavior pattern can be modelled to guide the automatic systems and to optimize the image interpretation models, for example by reducing the search space. Further, the computer can attempt to predict human intention and react accordingly. However, humans too are not always accurate. Consequently we envisage a tightly coupled, real-time, error-correcting interaction between human and machine in order to make map revision more efficient. To implement this we need to better understand these interactions.

Successful user modelling has been performed in several applications. Researchers at Microsoft Research reported their study of Bayesian user modelling for interring the goals and needs of software users in the Lumière project [60]. In this project, a special version of Excel was created to capture the mouse and keyboard actions as well as the status of data structure in Excel files. Using a specially developed language, these atomic events were transformed into variables and fed into the Bayesian models for temporal reasoning about the user actions.

Another project was reported on building a user support system by action-sequence

*A version of this chapter has been published in the Proceedings of the 10th International Workshop on Structural and Syntactic Pattern Recognition, pp. 287-295, Lisbon, Portugal, August 18-20 2004.

based user modeling [30]. This research has been integrated with two commercial software products. The user interactions with the software systems were captured and used to update the stored user knowledge in a global and a local model. The global model calculated the user's overall application system knowledge, while the local model calculated the user's expertise related to the specific tasks. Then the system provided the user with support according to the knowledge level.

Other work includes the mouse trajectory prediction system by Murata [100], the action sequence based user modelling system by Encarnacao and Stoev [30], the user action reasoning system by Virvou and Kabassi [132], the visual estimation model by Healey *et al.* [51], and the human attention modelling research by Horvitz *et al.* [61].

All these systems are not open environments. Thus, they cannot provide other researchers a platform and database for future study. Further, the user modelling presented in the above work are not in semi-automatic image interpretation. In the following sections, we introduce our work on user modelling in the USGS map revision environment in which user actions can be captured using open-source libraries. The user modelling is closely coupled with the image features in the aerial imagery.

5.2 Predicting Human Gaze with Hidden Markov Models

One way to reduce the human workload in map revision is to reduce drawing actions and view change actions. This can be achieved by predicting when humans are likely to change the view displayed on the screen, and what new view is likely to be selected. This allows the target image area to be prefetched and a view change to be performed automatically. It requires the modelling of the interconnection between human attention and actions. To do so one can use Markov model or Hidden Markov Models (HMMs) [113].

5.2.1 Hidden Markov Models

A hidden Markov model [113] is a finite set of states, $\mathbf{X} = \{x_1, x_2, \ldots, x_N\}$, where N is the number of states. The state at time t is defined as $\mathbf{x}_t$. Each state is associated with an initial probability

$$\pi_i = P[\mathbf{x}_1 = x_i] \tag{5.1}$$

Thus, $\pi = \{\pi_i\}$ defines the initial state distribution. The state transitions are governed by transition probability distribution $A = \{a_{ij}\}$ such that

$$a_{ij} = P(\mathbf{x}_{t+1} = x_j | \mathbf{x}_t = x_i) \tag{5.2}$$

At any given time t, an HMM can output observation $\mathbf{z}_t$ from a finite set of observations, $\mathbf{Z} = \{z_1, z_2, \ldots, z_M\}$, where M is the number of distinct observations. The state-conditional observation probability distribution is $B = \{b_i(k)\}$, such that

$$b_i(k) = P(\mathbf{z}_t = z_k | \mathbf{x}_t = x_i) \tag{5.3}$$

Using A, B, π, M and N, we can define an HMM as

$$\lambda = (A, B, \pi) \tag{5.4}$$

Given an observation sequence $\mathbf{z}_{1:t} = \{\mathbf{z}_i, i = 1, \ldots, t\}$, the state sequence $\mathbf{x}_{1:t} = \{\mathbf{x}_i, i = 1, \ldots, t\}$ is hidden, and only can be observed through a set of stochastic processes that produce the observation sequence. In general, there are three problems associated with HMMs [113]:

1. Find $P(\mathbf{z}_{1:t}|\lambda)$, the probability of the observation sequence given the model.
2. Find the most likely state sequence given $\mathbf{z}_{1:t}$ and λ.
3. Adjust λ to maximize $P(\mathbf{z}_{1:t}|\lambda)$.

Problems 1 and 2 are typically solved by Viterbi algorithm [133]. Problem 3 can be solved by Baum-Welch algorithm [113], a form of EM (Expectation-Maximization) algorithm.

Forward and Backward Operators

The Viterbi algorithm is a dynamic programming algorithm, in which a forward operator and a backward operator are used.

In the forward operator, the forward variable $\alpha_t(i)$ is defined as

$$\alpha_t(i) = P(\mathbf{z}_{1:t}, \mathbf{x}_t = x_i | \lambda) \tag{5.5}$$

$\alpha_t(i)$ can be solved using the iterative method in Algorithm 1.

In the backward operator, the backward variable $\beta_t(i)$ is defined as

$$\beta_t(i) = P(\mathbf{z}_{t+1:T} | \mathbf{x}_t = x_i, \lambda) \tag{5.6}$$

$\beta_t(i)$ can be solved in the same manner as $\alpha_t(i)$, however, in the reverse order, as shown in Algorithm 2.

Algorithm 1 Forward Operator

Input: The observation sequence $\mathbf{z}_{1:T}$, the HMM λ.

Initialize:

$$\alpha_1(i) = \pi_i b_i(\mathbf{z}_1) \qquad 1 \leq i \leq N \tag{5.7}$$

for $t = 1$ to $T - 1$ **do**

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(\mathbf{z}_{t+1}) \qquad 1 \leq j \leq N \tag{5.8}$$

end for

Output:

$$P(\mathbf{z}_{1:T}|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{5.9}$$

Algorithm 2 Backward Operator

Input: The observation sequence $\mathbf{z}_{1:T}$, the HMM λ.

Initialize:

$$\beta_T(i) = 1 \qquad 1 \leq i \leq N \tag{5.10}$$

for $t = T - 1$ to 1 **do**

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(\mathbf{z}_{t+1}) \beta_{t+1}(j) \qquad 1 \leq i \leq N \tag{5.11}$$

end for

Output:

$$P(\mathbf{z}_{1:T}|\lambda) = \sum_{i=1}^{N} \beta_1(i) \tag{5.12}$$

The Viterbi Algorithm

The purpose of the Viterbi algorithm is to find the most likely state sequence given the observation sequence and the HMM model. One solution is to calculate the maximum *a posteriori* (MAP) estimation the state sequence. At time t, we define:

$$\delta_t(i) = \max_{\mathbf{x}_{1:t-1}} p[\mathbf{x}_{1:t-1}, \mathbf{x}_t = i, \mathbf{z}_{1:t}|\lambda] \tag{5.13}$$

The optimal sequence can be extracted using Algorithm 3.

Algorithm 3 Viterbi Algorithm

Input: The observation sequence $\mathbf{z}_{1:T}$, the HMM λ.

Initialize:

for $i = 1$ to N **do**

$$\delta_1(i) = \pi_i b_i(\mathbf{z}_1) \tag{5.14}$$

$$\psi_1(i) = 0 \tag{5.15}$$

end for

for $t = 2$ to T **do**

for $j = 1$ to N **do**

$$\delta_t(j) = \max_i[\delta_{t-1}(i)a_{ij}]b_j(\mathbf{z}_t) \quad 1 \leq i \leq N \tag{5.16}$$

$$\psi_t(j) = \arg\max_i[\delta_{t-1}(i)a_{ij}] \quad 1 \leq i \leq N \tag{5.17}$$

end for

end for

$$p^* = \max_i[\delta_T(i)] \tag{5.18}$$

$$\mathbf{x}_T^* = \arg\max_i[\delta_T(i)] \tag{5.19}$$

for $t = T - 1$ to 1 **do**

Reconstructed state sequence

$$\mathbf{x}_t^* = \psi_{t+1}(\mathbf{x}_{t+1}^*) \tag{5.20}$$

end for

Output: The optimal state sequence $\mathbf{x}_{1:T}^*$

The Baum-Welch Algorithm

An initial estimation of an HMM can be obtained by randomly initializing the model parameters, or using statistics from training data. Given a sequence of observations, it is possible

to iteratively update the model parameters so that a maximum likelihood estimation of the HMM, $P(\mathbf{z}_{1:t}|\lambda)$, is obtained. The iterations in Baum-Welch algorithm [113] include two steps, calculating a re-estimated model, and substituting old parameters.

For the first step, we need to define the probability of being in state x_i at time t, and transit to state x_j at time $t+1$, given λ and $\mathbf{z}_{1:T}$:

$$\xi_t(i,j) = P(\mathbf{x}_t = x_i, \mathbf{x}_{t+1} = x_j|\mathbf{z}_{1:T}, \lambda) \tag{5.21}$$

Using the forward and backward variables defined in the previous section, this probability can be computed as:

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(\mathbf{z}_{t+1})\beta_{t+1}(j)}{P(\mathbf{z}_{1:T}|\lambda)} \tag{5.22}$$

Defining the probability of being in state x_i at time t, given λ and $\mathbf{z}_{1:T}$,

$$\gamma_t(i) = \sum_j \xi_t(i,j) \tag{5.23}$$

we can calculate the expected number of transitions from x_i as $\sum_{t=1}^{T-1} \gamma_t(i)$, and the expected number of transition from x_i to x_j as $\sum_{t=1}^{T-1} \xi_t(i,j)$. using these expectations, an HMM can be re-estimated using the following formula:

$$\bar{\pi}_i = \gamma_t(i) \tag{5.24}$$

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{5.25}$$

$$\bar{b}_j(k) = \frac{\sum_{t=1 \text{ s.t. } \mathbf{z}_t = z_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{5.26}$$

Then we can update the HMM using Algorithm 4.

Performance Evaluation

Depending on the application of HMMs, different evaluation criteria can be used, for example, the correct recognition rate in speech or optical character recognition, or the number of correct predictions of protein-coding regions in genome sequences processing. The evaluation method used in this thesis was proposed by Caelli *et al.* [17]. A trained HMM is used

Algorithm 4 Update old HMM model

Input: The HMM $\lambda = A, B, \pi$.

repeat

Compute $\bar{\pi}_i$, $\bar{a}_{ij}$ and $\bar{b}_j(k)$ using equations 5.24 to 5.26

Define re-estimated model as $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$

set λ to be $\bar{\lambda}$

until λ converges

Output: updated model λ

as an observation generator, then the generated observation sequences are compared to the true observation sequence.

The observation sequences generation is performed by randomly selecting states, state transitions, and observations, according to the model parameters, using Monte Carlo method. Hamming distances between the generated sequences and the true observation are computed, with a mean and a standard deviation of the distances obtained. These measures assess the trained HMM on the likelihood that an observation sequence would match predictions. Specifically, a Hamming distance d between a generated sequence $\mathbf{g}_{1:T}$, and a true observation sequence $\mathbf{z}_{1:T}$ is calculated as:

$$d(\mathbf{g}_{1:T}, \mathbf{z}_{1:T}) = \frac{\sum_{i=1}^{T} \Phi(\mathbf{g}_i, \mathbf{z}_i)}{T} \tag{5.27}$$

where

$$d(\mathbf{g}_i, \mathbf{z}_i) = \begin{cases} 1, & \text{iff } \mathbf{g}_i \equiv \mathbf{z}_i \\ 0, & \text{otherwise} \end{cases} \tag{5.28}$$

5.2.2 Human Gaze Prediction

Along with the parsed human action sequences, we can 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 4.2.2, and the semantic groups were obtained by clustering actions based on their functions as used in the RGR system, such as, for example, a group of actions for drawing transportation symbols or water body symbols. In the sematic case, the actions were divided into 6 groups (6 states). 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 case, as shown in Figure 5.1.

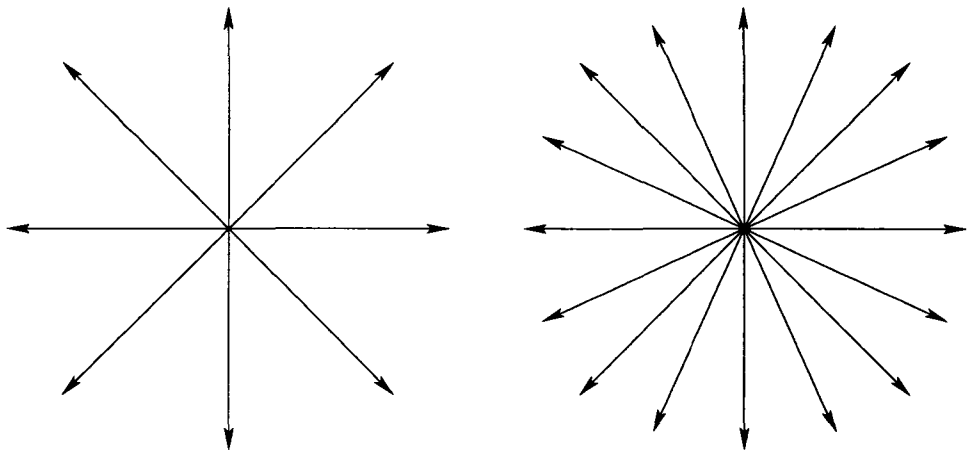


Figure 5.1: The directions of observation. Left graph shows 8 directions of gaze change. Right graph shows 16 directions of gaze change. When both are added with no change case, they correspond to 9 and 17 observations in an HMM.

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. Each task sequence contained 180 actions on average. These task sequences were further divided into shorter sequences with 2 actions each. Finally, we obtained 560 training sequences and 540 test sequences sampled at 1-second intervals, with an average length of 27 observations in each sequence.

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 5.1.

The numbers in Table 5.1 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, and 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 using 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

	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

Table 5.1: 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.

combination, the probabilities of correctly predicting the observation sequences, given the models and Viterbi solutions, were significantly above chance (last column of Table 5.1). These results are quite informative given the lengths of the sequences (27 on average) and prediction rates significantly above chance.

In real applications, however, predictions of viewing changes need to be much more accurate. Further, complete viewing prediction requires not only a direction, but also an exact location on the map. This suggests that a simple analysis of the user actions is not sufficient. A better prediction model should involve the recognition of features from image and maps.

Current research on 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 the need to pre-select points along the road [34]. It is normally difficult to extend these automatic methods in a robust and efficient manner to very large images, such as the DOQs of this project [40]. Our viewing prediction results provide the possibility of reducing the search area in a large image to relatively small areas in the predicted directions, so as to generate a semi-automatic road tracker. The size of the predicted area 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 based on the assumption that a small target area has been extracted. Then we compare the performance of human and computer in road tracking.

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

In semi-automatic systems, it is assumed that the human can perform tasks correctly and precisely. Further, what the computer determines as "incorrect" is 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.

A road segment is determined by two consecutive coordinates (mouse clicks), the axis joining the coordinates defines the human detected road. 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 [18]. As a result, points at maxima of gradient magnitude in the gradient direction were marked as edges, which may include both road and non-road edges, such as contour of cars. This edge operator was used because both straight and curved roadsides can be detected. Abrupt greylevel changes caused by surface material changes can also be detected and do not affect the extraction of candidate road edge points. Figure 5.2(a) and 5.2(b) show an example of the cropped image and the image after Canny edge detection.

(a) (b) (c)

Figure 5.2: (a) Cropped image from DOQ. (b) Human input (white blocks) and edges detected by Canny edge operator. (c) Axis detected by computer.

For each point on the axis defined by the human operator, we constructed a line perpendicular to the axis and determined the intersection points to the edges. These points were the candidates of the roadside. To reduce the influence of disturbances on the road, like cars and shadows, points on the edges with short length were removed from the candidate list. If two or more candidates were found, the road width limits defined by USGS were used as the upper and lower bounds of the distances between roadsides [128]. The two closest intersections to the axis detected by human that also met the width limit, were selected as the roadsides corresponding to the axis point. The connection of the mid-points of these

intersection pairs formed the axis detected by the computer. Figure 5.2(c) shows the result of Canny-edge axis detection of the image in Figure 5.2(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 had similar greylevels, the Canny operator failed to mark the road edges. To reduce this error, the roadsides were predicted by fitting a parabola to the most recent road points, as described in [93]. This error can also be avoided by detecting weak edges from gradient images using distance limits [138]. Another type of error came from disturbances on the road that had not been removed. Some of them were connected with the roadsides, which made the road appear 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.

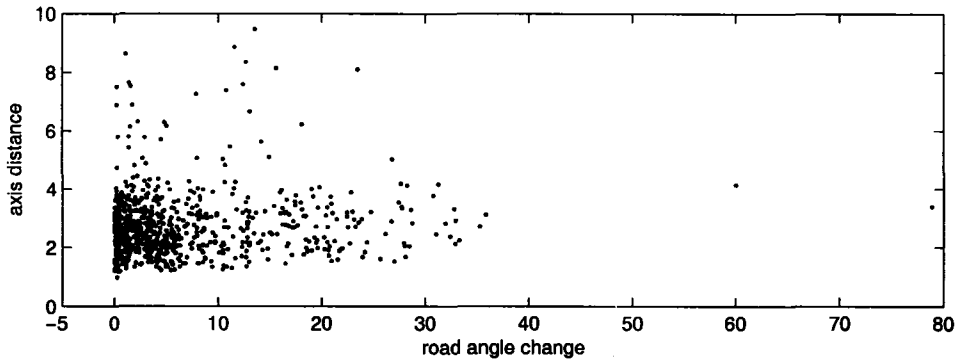


Figure 5.3: Distances of road axis versus road angle changes in the training sets.

To compare human and computer performance, the mean distance of the axis detected by computer and by human was calculated. Figure 5.3 shows the distribution of the distances versus road angle changes on the two training sets described in Section 5.2. The road angle change was obtained from the angle between two consecutive axes detected by human. We expected that the distances between the axes would increase along with the road angle changes, but the results show that there is no relationship between the two. In most cases, the distances between the axes 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) [129]. In the cases where the distances were too large to be acceptable, an analysis showed that although most errors were caused by the deficiency of the Canny-edge axis detection, some were caused by the human inaccuracy of in georeferencing ground object in DOQ with a map feature. The human input may lie 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 5.4.

(a) (b) (c)

Figure 5.4: (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.

5.4 Conclusion

We need to model user actions in order to develop semi-automatic image interpretation system for map revision. In this chapter, two experiments based on the user modeling were reported on predicting human viewing changes and on 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 more tightly coupled interactions between human and machine.

Chapter 6

Extracting Lines in Noisy Image Using Directional Information *

As we have discussed in Chapter 3, automatic image interpretation models are an important part of our human-computer interaction framework. A robust computational model can improve the utility of a system and the efficiency of the human-computer interactions. Thus, there is always a need for better models.

Line extraction is one of the major methods used for road detection. Having been performed for many years, it is still a hot topic in the image processing and computer vision research. One of the problems to solve in this topic is to extract linear features in noisy images. An application example is shown in Figure 6.1. There are two kinds of roads in this image: the wide, white one is a road for major traffic, while the thin, fuzzy lines are trails for oil and gas exploration. The trails are normally straight. The detection of these trails is very important for forest preservation and oil exploration planning, but the noisy background of a forest makes this task difficult. In this chapter, we introduce a linear feature extraction algorithm to tackle this problem.

6.1 Introduction

Line detection is a fundamental task in computer vision. The Hough transform is the best known method for detecting lines [66, 81]. Usually, the Hough transform starts with edge detection based on the local differential properties of the image. Then the edge map is transformed from image space into parameter space, where collinear points in the image space correspond to peaks in the parameter space. Candidate points for line segments in the image space can be restored from these peaks. Analyzing and grouping these candidate

*A version of this chapter will appear in the Proceedings of the 18th International Conference on Pattern Recognition, Hong Kong, China, August 20-24 2006.

points generates a final set of line segments.

Noise strongly affects line detection in the Hough transform. This is illustrated in the example in Figure 6.1(a), where the task is to find the four straight line segments that form a rectangle in an aerial photo. First, edge detection in this image is difficult because noise cannot be completely removed in the smoothing step. For example, the Canny edge detector [18], which often gives the best performance in edge detection, may not find a suitable scale to accurately detect and localize edges while efficiently removing the noise when the image is smoothed isotropically. This is shown in Figure 6.1(b) in which many edges are detected in the background forest area. Second, the noisy edges generate noisy peaks in the parameter space. These noisy peaks may be mixed with a butterfly pattern (see below) of transformed line segments, leading to the generation of false peaks. It is difficult to distinguish true peaks from false peaks, which, in turn, prohibits correct grouping of candidate points into true line segments.

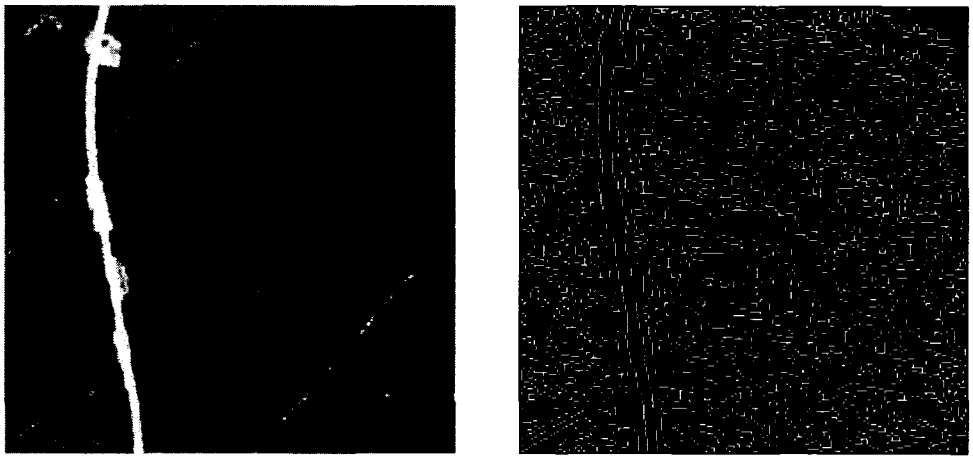


Figure 6.1: Line detection in noisy image. (a) An aerial photo of size 250 by 250 pixels with lines to be detected in noisy background. (b) The Canny edge detection result on (a).

Several methods have been proposed to solve these problems, such as anisotropic filtering in edge detection [109, 41], peak analysis in Hough transformation [131, 39], and grouping of directional features [19].

We propose a new method for effectively solving these problems using directional information. First, a Gabor filter is applied to the image to compute the dominant local orientation at each pixel. Then anisotropic Gaussian filtering is performed before edges are detected using the Canny edge detector. In this way, the noisy edges are reduced. Second,

in the Hough transform, peaks are calculated from the edges and line segments are restored. Then a peak selection algorithm based on directional information is used to distinguish true line segments from the line segments generated from spurious noisy edges.

6.2 Anisotropic Filtering for Edge Detection

The purpose of anisotropic filtering is to effectively reduce the influence of noise while accurately detecting edges. In isotropic filtering, noise is smoothed in the same way in all directions. To reduce the side-effect of noise along edges or lines, one can take advantage of the directional property of these linear features. This requires fine-tuning the filter in the direction of the lines. To obtain the dominant local orientation, we used a Gabor filter. Then the image was smoothed anisotropically using the orientation map, followed by edge detection using the Canny edge detector.

6.2.1 Estimating Local Orientation

The Gabor filter is a bandpass filter [26]. A 2-D Gabor filter is a Gaussian envelope modulated by a complex sinusoidal carrier. An even-symmetric Gabor filter is given by [59]:

$$g_t(x,y) = \exp(-(\frac{x^2}{2\sigma_x^2} + \frac{y^2}{2\sigma_y^2}))\cos(\frac{2\pi x}{t}) \tag{6.1}$$

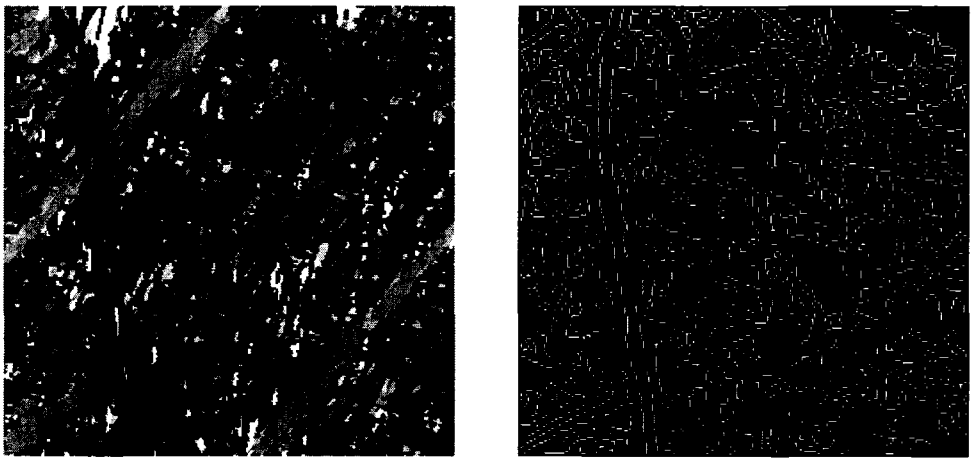
where t defines the wavelength of the Gabor filter, (σ_x, σ_y) defines the scale of the Gaussian envelope along the x-axis and y-axis respectively.

We may change the orientation of Gabor filter to angle θ, so that

$$g_{t,\theta}(x,y) = g_t(x',y') \tag{6.2}$$

where $x' = x\cos\theta + y\sin\theta$ and $y' = -x\sin\theta + y\cos\theta$.

To implement this even-symmetric Gabor filter, we need to determine parameters t, σ_x, σ_y, and θ. t is a parameter related to the edge frequency, and was set to 5, while σ_x, σ_y were both set to 3. The orientations θ were set to 16 orientations uniformly distributed in $[0,\pi)$. The input image was filtered iteratively on these orientations. The final orientation map was obtained by using the maximum response for each pixel over all orientations. An example of the extracted orientation map of Figure 6.1(a) is shown in Figure 6.2(a). The dominant orientation at each pixels is displayed in different greylevels and is used to guide the anisotropic filtering of the original image.



(a) (b)

Figure 6.2: Anisotropic filtering and edge detection. (a) Orientation map from Gabor filtering on figure 6.1(a). (b) Canny edge detection on anisotropically smoothed image.

6.2.2 Anisotropic Filtering

A fast anisotropic Gaussian filter has been proposed by Geusebroek and Smeulders [41]. The oriented filter in two dimensions is given by the convolution of two Gaussian filters:

$$g(u) = \frac{1}{\sqrt{2\pi}\sigma_u} \exp(-\frac{u^2}{2\sigma_u^2}) \tag{6.3}$$

and

$$g(v) = \frac{1}{\sqrt{2\pi}\sigma_v} \exp(-\frac{v^2}{2\sigma_v^2}) \tag{6.4}$$

where $u = x\cos\theta + y\sin\theta$ and $v = -x\sin\theta + y\cos\theta$. Here, θ is the orientation of the anisotropic Gaussian filter, x and y are the cartesian coordinates of the image pixels.

To facilitate the computation, the filter is transformed into image coordinates along the x-direction and t-direction, as shown in figure 6.3. The transformed filters are:

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma_x} \exp(-\frac{x^2}{2\sigma_x^2}) \tag{6.5}$$

and

$$g(t) = \frac{1}{\sqrt{2\pi}\sigma_t} \exp(-\frac{t^2}{2\sigma_t^2}) \tag{6.6}$$

where $t = x\cos\phi + y\sin\phi$. The relationship between σ_x, σ_ϕ and σ_u, σ_v, θ is:

$$\sigma_x = \frac{\sigma_u\sigma_v}{\sqrt{\sigma_v^2\cos^2\theta + \sigma_u^2\sin^2\theta}} \tag{6.7}$$

$$\sigma_\phi = \frac{1}{\sin\phi}\sqrt{\sigma_v^2\cos^2\theta + \sigma_u^2\sin^2\theta} \tag{6.8}$$

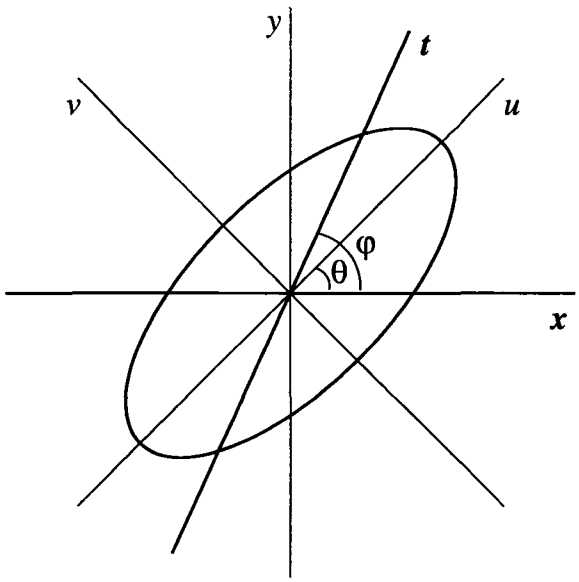


Figure 6.3: Anisotropic Gaussian filter.

$$\tan\phi = \frac{\sigma_v^2 \cos^2\theta + \sigma_u^2 \sin^2\theta}{(\sigma_u^2 - \sigma_v^2)\cos\theta \sin\theta} \tag{6.9}$$

In this way, a 2-D anisotropic Gaussian filter can be implemented as a convolution of two 1-D Gaussians in the x-direction and ϕ-direction.

6.2.3 Edge Detection

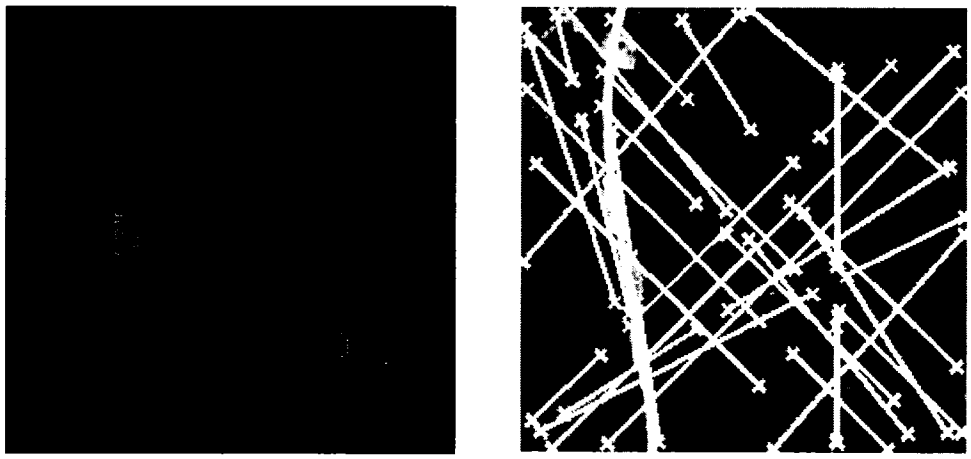
Edge detection was performed by a Canny edge detector to generate a binary edge map. However, in the Gaussian smoothing step, the traditional isotropic smoothing was replaced by the anisotropic filtering introduced above. The resulting edge map is shown in figure 6.2(b). Compared to Figure 6.1(b), the number of noisy edge points was reduced while the true edges were preserved. This is due to the fact that noise different from the dominant local orientation has been further smoothed out.

6.3 Line Detection using Hough Transform

The Hough transform is performed on an edge map to transform the $x - y$ coordinates of edge points into $\rho - \theta$ space. We used the normal parameterization proposed by Duda and Hart [27]. A line in an image is given by

$$\rho = x\cos\theta + y\sin\theta \tag{6.10}$$

where ρ is the distance of the line to the origin, and θ is the angle between the normal of the line and the $x-$axis, and (x, y) is any point on the line.



(a) (b)

Figure 6.4: Hough transform and problems. (a) Top 20 Hough peaks detected from image 6.2(b). (b) Lines (bounded by crosses) correspond to the peaks in (a) using traditional peak selection.

In the implementation of the Hough transform, the $\rho - \theta$ space is normally quantized into cells. Assuming that a line is mapped to a peak at point (ρ_0, θ_0), the collinear points on the line generate a pattern in the parameter space that is often referred to as a butterfly [131]. The analysis on the butterfly is supposed to be restricted to a small neighborhood window $(-w_\rho/2, w_\rho/2) \times (-w_\theta/2, w_\theta/2)$ around the peak (ρ_0, θ_0). The peaks in the neighborhood window can then be grouped and removed in a sequential peak selection [145].

One problem of this peak selection and grouping strategy is that noisy edge points may generate noisy peaks in the parameter space. For example, Figure 6.4(b) shows the lines restored from top 20 peaks in Figure 6.4(a), which was generated by Hough transform from Figure 6.2(b). Due to the noisy edge points, most extracted lines were false positive results and a true line in the lower-left corner was missing because it did not correspond to the top 20 peaks. Although we can increase the scale of the anisotropic Gaussian filter to further eliminate noise in edge detection, it usually cannot do so completely, and, as a side-effect, may also remove correct line points. The noisy peaks thus cannot be excluded without further analysis.

6.4 Peak Selection using Directional Information

An analysis of Figure 6.2 reveals that the orientation of line segments is helpful in peak selection. This is based on the fact that, in peaks mapped from lines, the collinear edge

points that contribute to the peaks tend to have the same local orientation, while the noisy edge points do not. This directional information, combined with traditional line grouping method in Hough transform, can help to distinguish true line segments from noisy ones. In this way, the peak selection can be considered as a classification by a decision tree. Each node in the decision tree classifies the lines mapped from a peak in the parameter space using a certain feature.

We developed the following algorithm for the peak selection using the Hough peaks (P) and the orientation map (O) as input, and generating true line segments as output:

Algorithm 5 Peak selection for line segment generation.

Peak Selection

Input: The Hough peaks $P = \{p_1, ..., p_n\}$, the orientation map O, thresholds t_1, t_2, t_3, t_4
Initialize: Sort P according to the intensity.
  for each peak p_i **do**
    Restore the contributing edge points EP_i
    $\sigma_{p_i} \leftarrow$ average deviation on directions of EP_i from O
    if $\sigma_{p_i} > t_1$ **then**
      Remove p_i and neighborhood peaks from P {*first step*}
    else
      Group points in EP_i with distance smaller than t_2 into line segments $LS_i = \{ls_1, ..., ls_m\}$
      $L_{LS_i} = \{l_1, ..., l_m\} \leftarrow$length of the line segments {*second step*}
      for Each line segment ls_j **do**
        if $l_j < t_3$ **then**
          Remove ls_j from LS_i {*third step*}
        end if
      end for
      $al_i \leftarrow$average length of LS_i
      if $al_i > t_4$ **then**
        Add LS_i into output {*fourth step*}
      end if
    end if
  end for
Output: Line segments

The first step in the algorithm is the direction judgement which enables the peak suppression and removal of most line segments composed of noisy edge points. The second and third step perform the point grouping, which excludes short noisy line segments. The last step removes noisy lines which are composed of short line segments that have not been successfully eliminated in the previous steps.

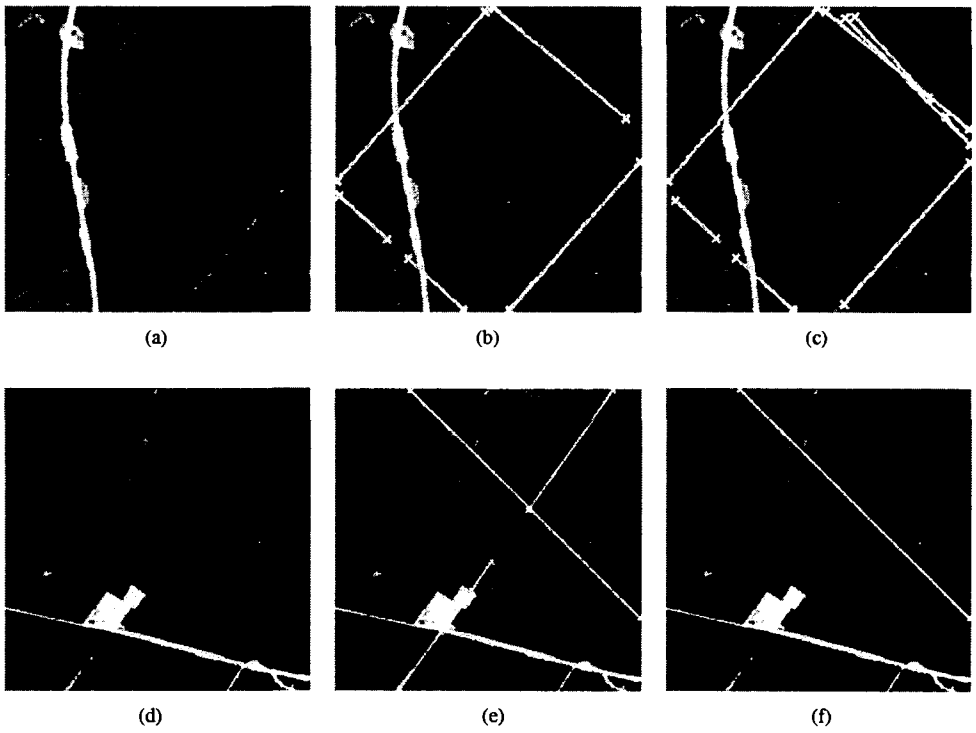


Figure 6.5: Line (bounded by crosses) detection results. (a) An image sample. (b) Lines detected on (a) using proposed method. (c) Lines detected using isotropic filtering. (d) Another image sample. (e) Lines detected on (d) using proposed method. (f) Lines detected using isotropic filtering.

6.5 Experimental Results

We performed experiments on sub-images extracted from a 7140 * 5940 aerial photo with ground resolution of 5 meters. The system went through all the steps described in the previous sections. The standard deviations of the anisotropic Gaussian filter were set to $\sigma_v = 1$ and $\sigma_u = 3\sigma_v$. The parameters of the Hough transform were set to $\Delta\rho = 1$ and $\Delta\theta = \pi/180$. The thresholds in the peak selection and line grouping were set to $t_1 = \pi/9$, $t_2 = 20$, $t_3 = 30$ and $t_4 = 40$, respectively. These are empirical values tuned for the target images.

The final result for Figure 6.1(a) is shown in Figure 6.5(b). The starting and ending points of the lines were marked by crosses. The results show that all target lines were detected successfully. Figure 6.5(e) shows another line detection result. Note that a short line segment is missing due to the weakness of the edges.

We compared the results of using isotropic and anisotropic filtering. In the former case, two noisy lines are detected, as illustrated in Figure 6.5(c). This is due to fact that too

many noisy edge points have been detected in Figure 6.1(b). These edge points can not be completely removed in the peak selection and thus generate false lines. In the latter case, three short lines are missing. This is also caused by the noisy edge points, which were removed in the peak selection due to their deviation to the dominant orientation of the line. The remaining edge points are not close enough to each other to be grouped into line segments.

6.6 Conclusion

This chapter introduced a line detection method based on directional information and Hough transform. The directional information is utilized in two stages. First, anisotropic filtering is used in the edge detection step. Each pixel in the image is smoothed according to their dominant local orientation extracted from an Gabor filter. Second, after the Hough transform, the peak selection and line segments grouping is guided by the same directional information. This method can successfully extract lines in noisy images.

Chapter 7

HCI-based Bayesian Filtering for Road Tracking in Aerial Imagery *

7.1 Introduction

Automatic road detection and tracking based on image interpretation has been one approach to speeding up the map revision process. It requires knowledge about the road database as well as image-related knowledge [24], including the road context, road characteristics, previous processing results, rules, and constraints [6]. A problem with such knowledge-based systems is that knowledge is pre-defined and fixed, whereas image road features vary considerably. The dynamics cannot be completely predicted, and they constitute the main source of problems with fully automated systems.

One solution to this problem is to adopt a semi-automatic approach that retains a "the human in the loop", where computer vision algorithms are used to assist humans performing these tasks [101, 110]. In this approach, dynamic knowledge can be transferred to computers, not only when necessary, but also to guide the computer. As reviewed in section 2.2, we have introduced several semi-automatic road tracking systems. These semi-automatic systems allow humans only to initiate the tracking process and/or to perform final editing. This makes the road tracking process difficult to control and leaves the combination of human and computer resources suboptimal. Typically, the tracking process is only guided by the most recent human input.

In this chapter, we present an approach that uses a semi-automatic road tracking system based on human-computer interaction and Bayesian filtering [4]. It is an application of the human-computer interaction framework proposed in Chapter 4. The Bayesian filters are

*A version of this chapter has been published in the Proceedings of the Workshop on Object Extraction for 3D City Models, Road Databases and Traffic Monitoring - Concepts, Algorithms, and Evaluation, pp. 35-40, Vienna, Austria, August 29-30, 2005.

used to estimate the current state of the system, based on past and current observations. When the Bayesian filters fail, a human operator observes the reason of the failure and initializes another filter. Observation profiles are generated from 2D features of the road texture, making the tracker more robust. Optimal profile matches are determined from the current state of the Bayesian filters and the multiple observations. The human operator interacts with the road tracker, not only at the beginning but throughout the tracking process. User input not only sets the initial state of the Bayesian filters but also reflects knowledge of road profiles. Consequently, the road tracker is more flexible in dealing with different kinds of road situations, including obstructions by vehicles, bridges, road surfaces changes and more.

The main contribution of the approach is to propose a general and robust system that effectively combines existing technology with task demands and human performance [50]. It is a practical solution to applications in remote sensing and image exploitation, where many automatic algorithms have been developed, but most of them have been unusable in reality [43].

7.2 System Overview

The application for our study is the USGS topographic map revision system introduced in Chapter 4. This platform uses the DOQ as the source of revision. An example of a road scene in a DOQ is shown in Figure 7.1. In the DOQ production, limited radiometric editing has been performed to the source aerial photographs in order to improve the image quality. When a DOQ is presented in the map revision system, no further image preprocessing is required to make the task easier for the human operator. For this reason, our system has been designed to not rely on further image preprocessing.

The task of map road layer revision is one where the principles in section 4 can be applied. In this case, the computer is trained to perform road feature tracking as consistent with experts as possible.

The road tracking process starts with an initial human input of a road segment, which indicates the road centerline. From this input, the computer learns relevant road information, such as starting location, direction, width, reference profile, and step size. This information is then used to set the initial state model and related parameters of the automatic road tracker. The computer also preprocesses the image to facilitate extraction of road features. The extracted features are compared with knowledge learned from the hu-

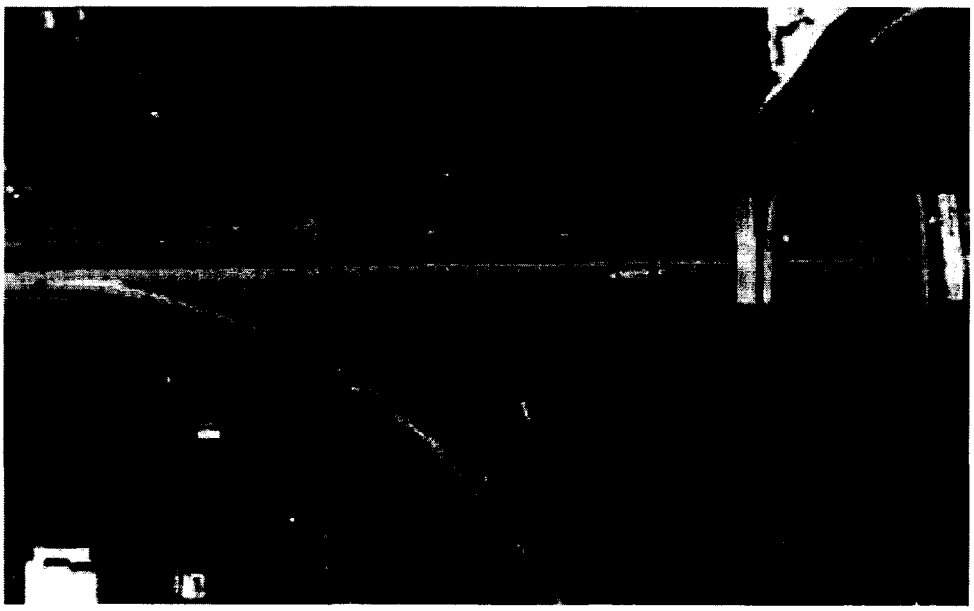


Figure 7.1: An image sample of size 663 by 423 pixels extracted from a DOQ.

man operator through profile matching. The computer tracks the road axis points using a Bayesian filter. Connecting these points forms the road centerline detected by the computer. This centerline can be automatically digitized on the map by calling the tool to draw road features. During tracking, the computer continuously updates road knowledge while, at the same time, evaluating the tracking results. When it detects a possible tracking problem or a tracking failure, it returns control back to the human. The human observes the road changes, diagnoses the failure reason and indicates the correct tracking direction by inputting a new road segment. The new input enables prompt and reliable correction of the state model of the tracker. The new segment can either be input at the current location to resume tracking, or it can be input at the location where the tracking error happened. In this way, errors can be identified and corrected in real time, without interrupting the tracking process. Figure 7.2 shows the architecture of the system.

This human-computer interaction determines what knowledge is to be passed between human and computer about the relevant road features in the image. This involves more formal methods for encoding road features and cartographical actions such as tracing roads, road boundaries and their related symbols. This involves defining expert static and dynamic knowledge on roads (see Section 4). Some useful static knowledge on roads is described in [5, 134]:

- Roads are elongated.

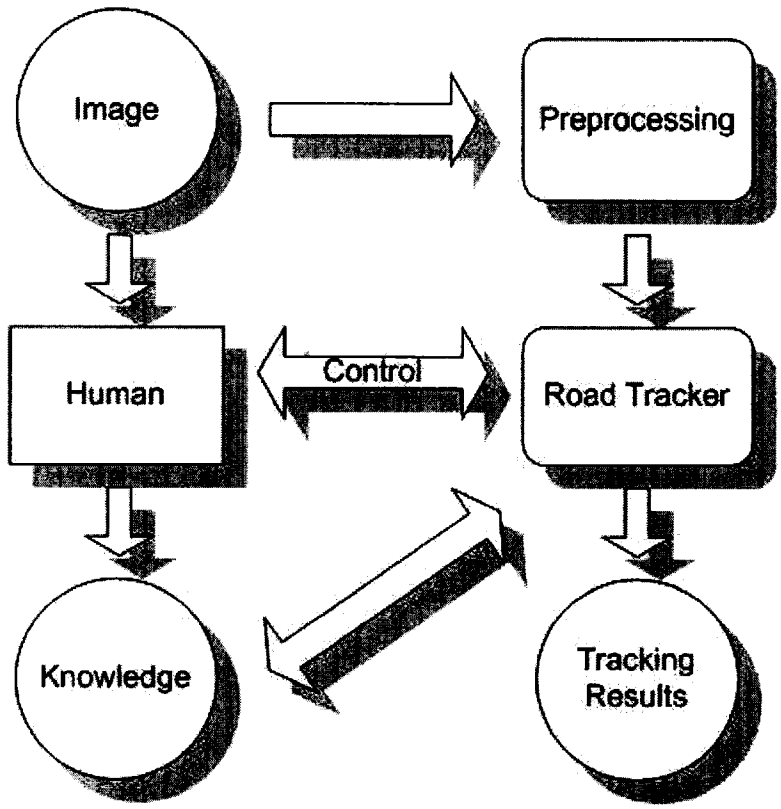


Figure 7.2: Block diagram of road tracking system. This system is composed of five components. 1. human; 2. computer vision algorithms to process images and track roads; 3. an interface to track and parse human actions; 4. a database to store knowledge, and 5. evaluation algorithms for feedback purposes, so that the computer can quantitatively evaluate human input and its own performance, and the human can evaluate the performance of computer.

- Road surfaces usually have high contrast with adjacent areas.
- Road surfaces are smooth and homogenous.
- Road curvatures have an upper bound.
- The width of roads is bounded. The upper and lower bounds of the width depend on the importance of the road.
- Roads are connected, networked entities.

Such static knowledge determines what road tracking method can be used. For example, the curvature property constrains the types of road position extrapolation methods that can be used to predict the road position [93]. The contrast and homogenous properties make it possible to use edge detection methods to detect parallel road edges [80, 10].

Dynamic properties define the expected changes of road features, for example, radiometric changes caused by different road materials, changes in contrast between road surface and adjacent areas caused by road texture, lighting conditions, and weather conditions, special properties of crossings, bridges, and ramps, road appearance changes caused by background objects such as cars, shadows, trees, and others.

Because dynamic properties cannot be predicted completely, human input is required to guide feature extraction and road tracking. We also need to explore alternative ways to quantize road features. Instead of traditional one-dimensional road features, we use two-dimensional road features that contain more radiometric information and might therefore be more robust.

7.3 Exploring The Roles of the Human and Computer in Road Tracking

The human operator is at the center of the proposed system. The operator affects the tracker in two ways. First, the operator gives the computer a starting location and direction of a road by initializing road segments. These inputs are used by the computer to extract reference profiles, to detect the road edges, and to estimate the road width. They are also used to set the state model of the Bayesian filter for tracking. Whenever the computer fails, the operator observes the road changes, diagnoses the failure reason, and indicates the corrected tracking direction. The new input enables prompt and reliable correction of the state model of the tracker. Second, reference profiles extracted from human inputs are stored, and the road tracker gradually accumulates knowledge on these reference profiles. These profiles represent different road situations that the tracker has not yet seen. This knowledge passing process makes the tracker increasingly robust.

The computer also accumulates knowledge by itself. During the tracking, it continues updating the matched reference profiles with the latest tracking results. This enables the tracker to adapt to gradual road changes, so that human inputs can be reduced.

The tracker performance is always evaluated in so far as, when there is lack of confidence over several consecutive positions, the system returns control to the human and waits for the next input. This evaluation is performed via cross-correlation, where new profiles are defined in terms of their lack of correlation with past ones. In this way, knowledge redundancy is avoided and the knowledge base does not expand too quickly, thus avoiding a reduction in tracking performance.

This intelligent tutor/decision maker and apprentice/assistant architecture provides a

useful communication path between human operator and the computer. The computer can learn quickly from humans, and it can work more and more independently as tracking goes on.

7.4 Preprocessing

The preprocessing module consists of three components, image smoothing, road width estimation, and extraction of an initial reference-profile.

7.4.1 Smoothing Step

In the smoothing step, the input image is convolved with a 5×5 Gaussian filter

$$G = \exp(-\frac{x^2 + y^2}{2\sigma^2}) \tag{7.1}$$

where $\sigma = \sqrt{2}$ pixels. This filter is used to set the analysis scale and to reduce high-frequency noise.

7.4.2 Road Width Estimation

Road width determines whether road profiles can be correctly extracted or not. If an estimated road width is smaller than the truth, road edges will not be completely included in the road profile. On the contrary, if the estimation is much larger than the truth, a lot of irrelevant information will be included in the road profile. In previous semi-automatic road trackers, the road width was typically entered by the human operator [93, 134, 9], whereas in our system, the road width is estimated automatically. A road segment is entered by the human operator with two consecutive mouse clicks with the axis joining the points defining the road center line. We assume that the roadsides are straight and parallel lines on both sides of the road axis. Road width can be estimated by calculating the distance between the roadsides. Further, knowledge about road characteristics also helps determining road edges because road width varies as a function of road class.

To detect the road edges, a method based on gradient profiles has been developed. This edge detector first estimates the true upper and lower bound of the road width, with the USGS road width definitions serving as a reference [128]. At each axis point a profile is extracted perpendicular to the axis. The length of the profile is bounded by the road width limits defined by USGS. The gradient of the profile along the profile direction is calculated and one point is selected on both sides of the axis point where the largest maximum gradient is found. If several equal largest local maxima are found, the first two local maxima are

used. The distance between the two points is considered the road width at this point. For a road axis segment, we obtain a probability density function $p(x)$

$$p(x_i) = \text{number of times } x_i \text{ appears}, \quad 1 \leq i \leq n \tag{7.2}$$

where x_i is the road width value extracted above. n depends on the road width limit from USGS and the complexity of the road conditions. Because the image resolution is 1 meter, x_i correspond to road width of approximate x_i meters. Searching for the mode of the distribution

$$p(x^*) = \arg\max_x \ p(x_i) \qquad 1 \leq i \leq n \tag{7.3}$$

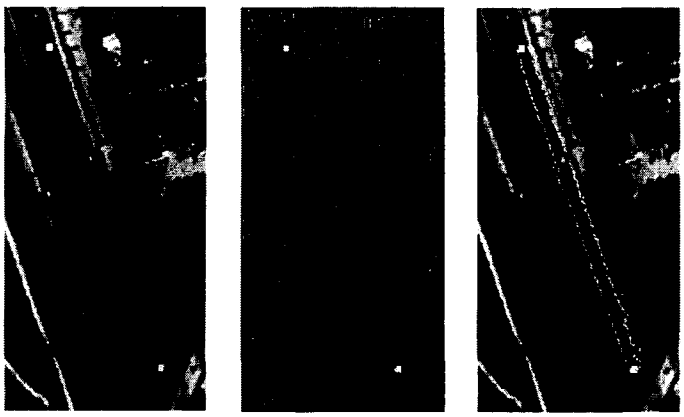
yields a dominant road width that appears most of the time. Then new road bounds are calculated using the functions

$$lb = x^* - e \ \ and \ \ ub = x^* + e \tag{7.4}$$

where lb is the new lower bound, ub is the new upper bound, and $e = 4$ is an empirical value that proved to be suitable for our application. Using the new bounds, the edge detector determines the new road width at each axis point and computes the average as the final road width for profile matching.

Figure 7.3 shows road edges detected by the Canny edge detector [18] and our own gradient-based detector. Since the Gaussian filter has already been applied to the image, the implementation of the Canny edge detector [18] starts from calculating the x- and y-gradient. Then the magnitude and direction of the gradient are calculated at each pixel. To perform the non-maximum suppression, we used empirical values 0.1 and 0.3 as low and high thresholds to determine the strong edges. Notice that the Canny edge detector does not take advantage of the known road direction and the road width limits, multiple edges may be detected, which causes trouble in finding the true road edges. Thus, our gradient profile based edge detector performs better than the Canny operator, at least in this specific application.

The mean value of the estimated road width was 10.8 pixels, with a standard deviation of 4.3 pixels. In 93.8% of the cases, the estimated road width varied between 6 to 18 pixels, depending on the real road width, the road conditions, and the locations of human inputs. The estimation of road width can be affected by several factors. First, pavement markings in multi-lane roads, rather than the true road edges, may generate the maximum gradient value. In our application, the aerial images had a resolution of 1 meter per pixel. Thus, pavement markings were either not wide enough to be displayed or appeared to be less



(a) (b) (c)

Figure 7.3: Road edge detection results. (a) Cropped image from DOQ with human input (white blocks). (b) Result of Canny edge detector: note the presence of multiple road edges. (c) Result of gradient profile based detector: only one pair of road edges is detected.

salient after the smoothing step. Second, off-road areas and the road can be made of the same material, or have the same radiometric properties. For example, both the road and the sidewalk could be concrete. In this case, the maximum gradient is not found at the road edge. Our gradient based method either takes the first point at both sides of the road axis as the road edges, or it takes the edge of sidewalk as the road edge. In both situations, the road width is bounded by the limits defined by the USGS, so that it does not deviate far from the ground truth.

The reason for estimating the road width automatically was to allow the operator to focus on the road axis points and road directions, consistent with the operation of plotting roads in real-world map revision systems. However, it should be pointed out that tools with manual input are more accurate, though more time consuming, for estimating road widths, as used, for example in the ROADMAP system developed by [50].

7.4.3 Profile Extraction

An initial reference profile is extracted as a vector of greylevels from the road segment entered by the human operator. Later, new profiles are extracted from new human inputs and placed into a profile list for further use.

To improve robustness of the system, we use two-dimensional road features, i.e. in addition to searching along a line perpendicular to the road direction, we also search a line along the road direction. Profiles are extracted in both directions and combined. The parallel profile is useful since greylevel values vary little along the road direction, whereas

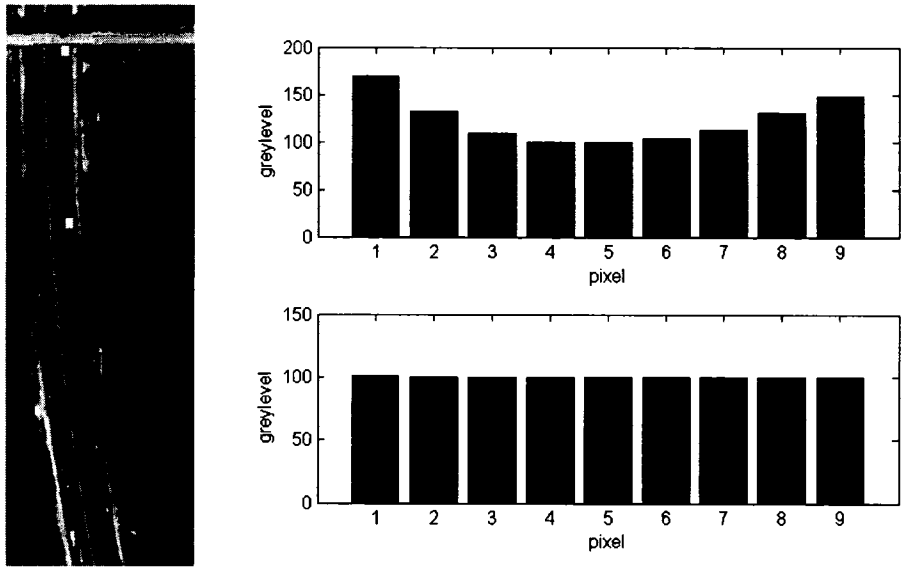


Figure 7.4: Profiles of an road segment. In the left image, two white dots indicates the starting and ending points of road segment input by human. The right graphs shows the road profiles perpendicular to (upper) and along (lower) the road direction.

this is not the case in off-road areas. Thus the risk of off-road tracking is reduced and, in turn, tracking errors are reduced. As will be described later in section 7.5.2, the observation profiles are also 2D features. An example of road profile extraction is shown in Figure 7.4

From each human input, we obtain a profile sequence that contains the road surface texture information which may include occluding objects. For a sequence of road profiles $P = [p_1, p_2, ..., p_n]$, profile extraction proceeds as follows. First, an average profile is calculated. Then each profile in the sequence is cross-correlated with the average profile. Whenever the correlation coefficient is below a threshold (an empirical value set to 0.8), the profile is removed from the sequence. In this way, all axis points are evaluated and road profiles extracted from noisy axis points, for example, where cars and trucks are presented, are removed. The algorithm iterates through all the profiles until a new profile sequence is generated, and the average profile of the new sequence is taken as the final road segment profile. The profile extraction algorithm is presented in Algorithm 6.

The effectiveness of this noise removal method is affected by road conditions. When occlusions are sparse, the method is quite effective. However, in the case of populated roads, *e.g.* on roads with a traffic jam, or roads under the shadow of trees, noisy reference profiles may be generated. In these cases, the performance of the system drops.

Algorithm 6 Reference profile extraction algorithm

Reference Profile Extraction

Input: A sequence of road profiles $P = [p_1 p_2 ... p_n]$

$\bar{p} \leftarrow \frac{\sum_{i=1}^{n} p_i}{n}$

for each p_i **do**

$r(p_i, \bar{p}) \leftarrow$ correlation coefficient of p_i and $\bar{p}$

if $r(p_i, \bar{p})$ is smaller than a threshold **then**

$P \leftarrow P - p_i$

end if

end for

$\bar{p} = \frac{\sum_{i=1}^{m} p_i}{m}$

Output: $\bar{p}$

7.5 Road Tracking

If we consider the road tracking process as a time series, it can be modelled by a state-space approach involving state evolution and noisy measurements. The state evolution of the tracking process can be defined as

$$\mathbf{x}_k = \mathbf{f}_k(\mathbf{x}_{k-1}, \mathbf{v}_{k-1}) \qquad k \in \mathbb{N} \tag{7.5}$$

where $\mathbf{x}_k$ is the state vector at time k, $\mathbf{v}_k$ is the process noise, and $\mathbf{f}_k$ is a function of $\mathbf{x}_{k-1}$ and $\mathbf{v}_{k-1}$.

Given an observation sequence $\mathbf{z}_{1:k}$, the tracker recursively estimates $\mathbf{x}_k$ using the prior probability density function $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ and the posterior probability density function $p(\mathbf{x}_k|\mathbf{z}_{1:k})$. The relationship between observations and states is defined by

$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{n}_k) \qquad k \in \mathbb{N} \tag{7.6}$$

where $\mathbf{n}_k$ is the measurement noise.

Depending on the properties of the state evolution, the observations, and the posterior density, the tracking problem can be solved with different approaches, such as Kalman filters, hidden Markov models, extended Kalman filters and particle filters [70, 113, 137, 4].

7.5.1 State Model

Road axis points are tracked using recursive estimation following [134], who proposed the following state model:

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \theta \\ \theta' \end{bmatrix} \tag{7.7}$$

where x and y are the coordinates of road axis points, θ is the direction of the road, and θ' is the change in road direction. The state model is updated by the following non-linear function

$$\mathbf{x}_k = \begin{bmatrix} x_{k-1} + \tau \cos(\theta_{k-1} + \tau \frac{\theta'_{k-1}}{2}) \\ y_{k-1} + \tau \sin(\theta_{k-1} + \tau \frac{\theta'_{k-1}}{2}) \\ \theta_{k-1} + \tau \theta'_{k-1} \\ \theta'_{k-1} \end{bmatrix} \tag{7.8}$$

Differences between this simplified process and the true road shape are interpreted as process noise $\mathbf{v}_k$, whose covariance matrix is Q_k.

In Equation (7.8), τ is the interval between time $k-1$ and k, determining the distance the road tracker traverses in each step. Initially it is set to the length of the road width and it is affected by three parameters. The first parameter is a "jump-over" factor corresponding to the internal evaluation of the road tracker (see Section 8.4). The second parameter corresponds to the prediction scale (see Section 7.5.6). The third parameter corresponds to the curvature of the road. When the road curvature is high, a smaller τ is used to avoid off-road tracking.

7.5.2 Observation Model

Observations are obtained by matching the reference profiles to the observed profiles, the latter being extracted in 2D at the position estimated by the state models. To minimize disturbances due to background objects on the road and road surfaces changes, a heuristic multiple-observations method is used to search the neighborhood of the estimated points for better matches. Euclidean distances between the matching and observed profiles are calculated, and the position with the minimum distance is selected as the optimal observation in an iteration. The observations $\mathbf{z}_k$ are thus calculated as

$$\mathbf{z}_k = \begin{bmatrix} x_k - s_k \, sin(\theta_k + \alpha_k) \\ y_k + s_k \, cos(\theta_k + \alpha_k) \end{bmatrix}, \tag{7.9}$$

where s_k is a shift from the estimated road axis point and α_k is a small change to the estimated road direction.

7.5.3 Extended Kalman Filtering

We have defined a tracking system based on nonlinear state and observation models. Extended Kalman filtering has been widely used to solve such nonlinear time series [16, 137] where the posterior density is assumed to be approximately Gaussian.

The tracking task is performed by estimating the optimal state $\hat{x}$ at each iteration. First, we compute Φ, which contains the coefficients of the linearized time update equations

$$\Phi_k = \frac{df_k(x)}{dx}|_{x=\hat{x}_{k-1}}. \tag{7.10}$$

The covariance matrix of the predicted state vector becomes

$$P_{k|k-1} = \Phi_k P_{k-1|k-1} \Phi_k^T + Q_{k-1}. \tag{7.11}$$

After the state update in Equation 7.8, the extended Kalman filter continues the iteration by solving the following measurement update equations:

$$K_k = P_{k|k-1} A^T (A P_{k|k-1} A^T + R_k)^{-1} \tag{7.12}$$

$$\hat{x}_k = \Phi_k \hat{x}_{k-1} + K_k(\mathbf{z}_k - A\Phi_k \hat{x}_{k-1}) \tag{7.13}$$

$$P_{k|k} = (I - K_k A) P_{k|k-1} \tag{7.14}$$

In equation (7.12), A is the measurement matrix

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}, \tag{7.15}$$

and R is the covariance matrix of the measurement noise

$$R_k = \sigma^2 \begin{pmatrix} sin^2(\theta_k) & sin(\theta_k)cos(\theta_k) \\ sin(\theta_k)cos(\theta_k) & cos^2(\theta_k) \end{pmatrix}, \tag{7.16}$$

where σ^2 is the variance of the shift s in the observation model. In the measurement update step, the Kalman gain is calculated in Equation 7.12. Then the predicted state and its covariance matrix are updated in Equations 7.13 and 7.14, respectively.

The initial state of the Extended Kalman filter is set to $\hat{x_0} = [x_0\ y_0\ \theta_0\ 0]^T$, where x_0 and y_0 are the coordinates of the end point of the road segment input by human operator, and θ_0 indicates the direction of the road segment. Starting from the initial state, the extended Kalman filter tracks the road axis points iteratively until x or y are outside the image boundaries or a stopping condition has been met.

Vosselman and Knecht suggested that the covariance matrix Q_k of the process noise in road tracking is mainly determined by the difference between the constant road curvature assumption and the actual curvature changes [134]. They set the standard deviation of the process noise in θ'_k to $1/400$ the radius of the road and propagate it to the standard deviation of other state variables. We followed this rule in determining the process noise.

7.5.4 Particle Filtering

Particle filtering, specifically the CONDENSATION algorithm proposed in [67], has been successfully used in modelling non-linear and non-Gaussian processes [4, 123, 82]. The filter approximates the posterior density $p(\mathbf{x}_k|\mathbf{z}_k)$ by the particle set $\{\mathbf{s}_k^i, w_k^i,\ i = 1, ..., N\}$ in each time step k, where w_k^i is a weight used to characterize the probability of the particle $\mathbf{s}_k^i$.

Given the particle set $\{\mathbf{s}_{k-1}^i, w_{k-1}^i,\ i = 1, ..., N\}$ at time $k-1$, the iteration k of the particle filter can be summarized as follows:

1. Construct cumulative density functions $\{c_{k-1}^i\}$ on the current particle set. Sample N particles $\{\mathbf{x}_{k-1}^j,\ j = 1, ..., N\}$ according to the cumulative density function. The sampling of the jth particle $\mathbf{x}_{k-1}^j$ is done by generating a uniform random number u^j on $[0, 1]$ and searching for the first particle $\mathbf{s}_{k-1}^i$ with $c_{k-1}^i \geq u^j$.

2. Update each particle by equation (7.8) to generate new particles $\{\mathbf{x}_k^j,\ j = 1, ..., N\}$. In the state update, the road curvature parameter θ' is influenced by a zero mean Gaussian random variable with unit variance.

3. Calculate new weights for each particle based on how well they fit the observation $\mathbf{z}_k$. The weights are normalized and are proportional to the likelihood $p(\mathbf{z}_k|\mathbf{x}_k^j)$. In this way, a new particle set $\{\mathbf{s}_k^i, w_k^i,\ i = 1, ..., N\}$ is constructed.

The estimated state at time k is then

$$E(\mathbf{x}_k) = \sum_{i=1}^{N} w_k^i \mathbf{s}_k^i. \tag{7.17}$$

In our application, we assume that at each time step, the state of the road tracking is supported by a set of particles. The number of particles is set to 20 times the road width in pixels, and the initial density of $p(\mathbf{x}_0)$ is set to a uniform distribution, which means each particle has the same initial probability. The particle filter gradually adjusts the weights of each particle during the evolution process. We further assume that the observation is normally distributed with standard deviation $\sigma = \sqrt{2}$ so the likelihood of the observation is

$$p(\mathbf{z}|\mathbf{x}^j) \propto \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{d_j^2}{2\sigma^2}), \tag{7.18}$$

where d_j is the Euclidean distance between the position of particle $\mathbf{x}^j$ and the observation.

7.5.5 Stopping Criteria

A matching profile is extracted from the observation model and cross-correlated with the reference profile. If the correlation coefficient exceeds some thresholds (*e.g.* 0.8 in [134]), the observation is accepted; if the coefficient is below the threshold, and some other conditions are met (*e.g.* a high contrast between the profiles), the observation is rejected. In this case, the Bayesian filters make another state update based on the previous state, using a larger time interval τ, so that the estimated position without accepted observation is jumped over. When contiguous jumps occur (set to 5 jumps), the Bayesian filter recognizes this as a tracking failure and returns control back to the human operator.

The jump-over strategy is particularly useful in dealing with small occlusions on the road, for example, when cars and long trucks are present. In these cases, the profile matching will not generate high correlation coefficient at the predicted state. The jump over strategy uses an incremented time interval to skip these road positions, so that a state without occlusions can be reached.

In real applications, however, road characteristics are more complex. Cross-correlation may not always generate a meaningful profile match, which in turn may lead to errors in the tracking process. For example, a constant road profile may generate high coefficient when cross-correlated with a profile extracted from an off-road area with constant greylevel. Furthermore, the Bayesian filters may often fail because the predicted position may not contain an observation profile that matches the reference profile. For example, when occlusions are present on the road, the reference and observation profiles may generate a small correlation coefficient, and leading to a rejection of the observation. The system then requires substantial interactions with the human operator, making the tracking process less efficient and quite annoying for the user.

7.5.6 Improving Efficiency

In previous algorithms [134, 9], each time a new reference profile was extracted, the old reference profile was discarded. In our system all the reference profiles are retained, and the road tracker gradually accumulates knowledge on road conditions. In profile matching, the latest profile is given the highest priority. When matching fails, the Bayesian filters search the list of reference profiles for a match. To reflect the gradual change of the road texture, the reference profile is updated by successful matches using a weighted sum. We call this the multiple profiles method.

We developed an algorithm to search for the optimal observation-reference profile com-

bination. The search space $V =< X, Y, \Theta >$ is defined by the current state $\mathbf{x_k}$, where X, Y and Θ are bounded by a small neighborhood of x, y and θ respectively. The search algorithm is described in Algorithm 7.

Algorithm 7 Searching algorithm for the optimal observation-reference profile combination

Searching Algorithm

Input: A list of reference profiles $P = p_1, p_2, ..., p_n$, a list of observation $V = v_1, v_2, \ldots, v_m$

for each $v_i \in V$ **do**
  extract profile p_i' at v_i
  $c(p_i', p_1) \leftarrow$cross-correlation coefficient of p_i' and p_1
end for
$c^* = \max(c(p_i', p_1))$
if $c^* > 0.9$ **then**
  update p_1
  return v^*
else
  for each p_i' **do**
    for each $p_j \in P, j \neq 1$ **do**
      $c(p_i', p_j) \leftarrow$cross-correlation coefficient of p_i' and p_j
    end for
  end for
  $c^* = \max(c(p_i', p_j))$
  if $c^* > 0.9$ **then**
    $p^* = p_j$ corresponding to c^*
    switch p_1 and p^*
    return v^*
  end if
end if

Output: the optimal observation v^*

In many tasks, humans use multi-scale attention to focus on important features and to reduce the influence of distractors [79]. To simulate such behavior, we adopted a step prediction scaling strategy to improve the efficiency of road tracking. A prediction scale is added to the state update model of the Bayesian filters, contributing to the calculation of the time interval τ. The initial prediction scale is set to 1. When a successful match happens the scale parameter is incremented by 1. Whenever matching fails the prediction scale is reset to 1. In this way, the time interval is adjusted automatically. If the road is long, straight and homogenous in surface, the road tracker can predict the next road axis point using a larger scale and ignore many details on the road, thus increasing the speed of the tracking process.

Table 7.1: Statistics on human input

	user1	user2	user3	user4	user5	user6	user7	user8
total number of inputs	510	415	419	849	419	583	492	484
total time (in seconds)	2765	2784	1050	2481	1558	1966	1576	1552
average number of inputs per task	18.2	15.2	15.0	30.3	15.0	20.8	17.6	17.3
average time per task (in seconds)	98.8	99.4	37.5	88.6	55.6	70.2	56.3	55.4

7.6 Experimental Results

7.6.1 Data Collection and Statistics

Eight students were required to plot roads manually in the USGS map revision environment, which displays the old map and the latest DOQ simultaneously on the screen. Plotting was performed by selecting tools for specific road classes, followed by mouse clicks on the perceived road axis points in the image. Before performing the actual annotation, each user was given 30 minutes to understand the road interpretation process as well as operations such as file input, road plotting, viewing change, and error correction. They did so by working on a real map for the Lake Jackson area in Florida. When they felt confident in using the tools and road recognition, they were assigned 28 tasks to plot roads on the map for the Marietta area in Florida. The users were told that road plotting should be as accurate as possible, *i.e.* the mouse clicks should be on the true road axis points. Furthermore, the road should be smooth, *i.e.* abrupt changes in directions should be avoided and no zigzags should occur. Although professional cartographers would be expected to perform such tasks better than the students used here, considering the simplicity of tasks, we believe the performance of students was close to that of experts. Indeed all users became familiar with the annotation operations in less than 15 minutes.

The plotting tasks included a variety of scenes such as trans-national highways, intra-state highways and roads for local transportation. Further, these tasks contained different road types such as straight roads, curves, ramps, crossings, and bridges. They also included various road conditions such as occlusions by vehicles, trees, or shadows.

Both spatial and temporal information on human inputs were recorded and parsed and only road tracking inputs were kept. We obtained 8 data sets, each containing 28 sequences of road axis coordinates tracked by a user. Table 7.1 shows some statistics on the human data, including the total number of inputs, the total time for road annotation, and the average time per input. The time for road annotation includes the time that users spent on image interpretation, plotting, and error correction. As shown in the evaluation criteria, these were

all taken into account in the efficiency calculation.

The number of inputs reflects how close the user zoomed in the image. When the image is zoomed in, mouse clicks traverse the same distance on the screen but correspond to shorter distances in the image. Thus, the user needs to input more road segments. The average time per input reflects the time that users require to detect one road axis and annotate it.

From the statistics, it is obvious that the users had performed the tasks in different patterns, which influenced the quality of the input. For example, more inputs were recorded for user 4 than for the other users. This was because user 4 zoomed the image into more detail than the other users. This made it possible to detect road axis locations more accurately in the detailed image. Another example is that of user 3, who spent much less time per input than the others. This was either because he was faster at detection than the others, or because he performed the annotation with less care.

7.6.2 Evaluation

The proposed system allows us to simulate the human-computer interactions using the recorded human data as virtual users. The road trackers interacts with the virtual users throughout the semi-automatic tracking process. In a road tracking task, the first two mouse clicks in the human data are used to generate a reference road profile. Then the road tracker tracks the road automatically until a failure happens. At the location of the failure, a new human input can be obtained from the virtual user to initialize a new tracking iteration. This process continues until the tracking task is finished. For a set of human data, we can apply different automatic tracking models, *e.g.* the Kalman filtering and particle filtering models. Finally, we can compare the performance between the simulated semi-automatic road tracker and the complete manual tracking for each user. Semi-automatic systems can be evaluated in many ways. For a real-world application, it may include user experience evaluation, as reported by [50]. Since our system is still in the simulation stage, we focused on the engineering aspect of the evaluation where the human factors components are only part of the assessment criteria. The criteria used to evaluate this system included the following:

- Correctness: Were there any tracking errors?
- Completeness: Were there any missing road segments?
- Efficiency: How much could tracking save in terms of human input, tracking distance, and plotting time?

Table 7.2: Comparison of road tracking results between particle filters and extended Kalman filter.

	input saving (%)	time saving (%)	distance saving (%)	RMSE (pixels)
extended Kalman filter	71.9	63.7	85.3	1.86
particle filter (average)	72.3	62.0	85.6	1.90
particle filter (best)	79.1	71.6	87.9	2.19

- Accuracy: How much did tracking deviate from manual inputs?

Correctness and completeness have the highest priority in Cartography. When errors occur, the human operator has to search and correct these errors, and this may take longer than the time that was initially saved. The same problem can occur if the update on a road is incomplete. The most important advantage of the proposed system over fully automatic ones is that the human involvement guarantees correctness and completeness of road tracking. The human operator always follows and interacts with the road tracker, and whenever an error happens, the operator can correct it immediately by initializing a new tracking iteration. The tracking process does not stop until the user decides that all roads have been recorded.

Consequently, in evaluating efficiency, savings in human inputs, in plotting time and in tracking distance have to be considered. The number of human inputs and plotting time are related and so reducing the number of human inputs also decreases plotting time. Given an average time for a human input, which includes the time for observation, plotting and the switching time between the two, we obtain an empirical function for calculating the time cost of the road tracker:

$$t_c = t_t + \lambda n_h. \tag{7.19}$$

where t_c is the total time cost, t_t is the tracking time used by road tracker, n_h is the number of human inputs required during the tracking, and λ is an user-specific variable, which is calculated as the average time for an input

$$\lambda_i = \frac{\text{total time for user i}}{\text{total number of inputs for user i}} \quad 1 \leq i \leq 8 \tag{7.20}$$

In real applications, the λ should be different, depending on the usability of the human-computer interface provided to the user. The savings in tracking distance is defined as the percentage of roads tracked by computer. Tracking accuracy is evaluated as the root mean square error between the road tracker and human input.

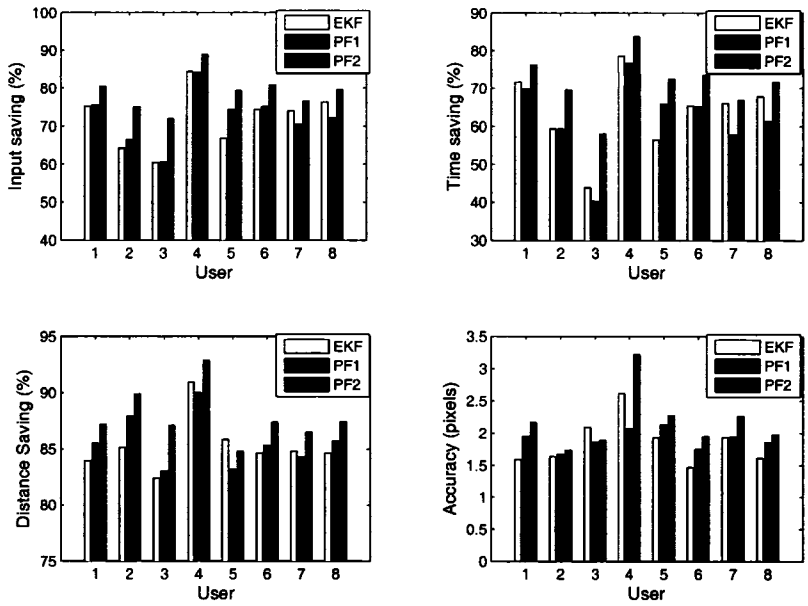


Figure 7.5: Comparison of tracking performances for the particle filters (PF1 and PF2) and the extended Kalman filter (EKF). PF1 shows the average performance of the particle filter over 10 Monte Carlo trials. PF2 shows the best performance of the particle filter over 10 Monte Carlo trials. The performance is evaluated on the saving of number of human inputs (upper left graph), the saving of total plotting time (upper right graph), the saving of tracking distance (lower left graph), and the accuracy as the root mean square error of the tracking results against the human input road axis (lower right graph).

7.6.3 Experimental Results

We compared the performance of the particle filter with the extended Kalman filter. Due to the factored sampling involved in the particle filter, the tracker may perform differently for each Monte Carlo trial. For this reason we evaluated the particle filter over 10 Monte Carlo trials and we report both average and best performance. Table 7.2 shows the performance comparison for the road trackers based on particle filtering and extended Kalman filtering. The proposed road tracking system shows substantial efficiency improvement in both non-linear filtering algorithms compared to a human doing the tasks alone. More detailed performance comparison for each user are shown in Figure 7.5.

In the proposed road tracking application, the system states and observations are subject to noise from different sources, including those caused by the image generation, disturbances on the road surface, road curvature changes, as well as other unknown sources. The nonlinear state evolution process propagates the noise into the state probability density

function (pdf). For this reason, it is better to construct a non-Gaussian, multi-modal pdf, and the extended Kalman filter and the particle filter are two sub-optimal methods to solve such systems. The experimental results shows that the performance of the extended Kalman filter and the average performance of particle filter are quite similar. Due to the uncertainty in the state evolution of particle filters, different pdfs of the states can be approximated in different Monte Carlo trials. When the approximation of the state pdf is a better fit to the true pdf, the particle filter out-performs the extended Kalman filter. This is why the best performance of the particle filter is better than that of the extended Kalman filter. We also noticed the compensation on the accuracy in particle filter tracking. This is caused by the error correction function of the particle filters, which tolerates more deviations in tracking.

Notice in Table 7.2 that both Bayesian filters provide approximately the same level of improvement. This suggests that a combination of both filters may further improve the performance of the tracking system. For example, both filters could perform the tracking task simultaneously using a two-filter competing strategy. A dynamic programming approach could be used to coordinate the behavior of the trackers, using the correlation traces to decide the optimal tracking path. The correlations could also provide human operators with realtime feedback on the "level of confidence" the system has in the prediction step. This could assist the human operator in monitoring the tracking and in making a decision whether to allow the system to continue tracking.

As can be seen in Figure 7.5, the Kalman and particle filters perform differently for different users suggesting that an adaptive, competitive filter selection strategy should be included in the complete tracking model.

The performance of the system also reflects the quality of human input. Input quality determines how well the template road profiles can be extracted. When an input road axis deviates from the true road axis, the corresponding template profile may include off-road content perpendicular to the road direction. Moreover, the profile along the road direction may no more be constant. Thus, the road tracker may not find a match between observations and template profiles, which in turn requires more human inputs, reducing the system efficiency.

Figure 7.6 shows a comparison of system with and without processing of human input during road template profile extraction. The tracking method is Kalman filtering. When human input processing is skipped, noisy template profiles enter the knowledge base. This increases the time for profile matching during the observation step of the Bayesian filter, which, in turn, causes the system efficiency to drop dramatically.

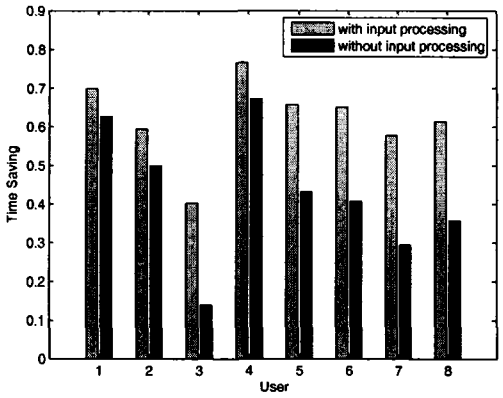


Figure 7.6: Efficiency comparison of semi-automatic road tracking with Kalman filtering.

Some tracking results are illustrated in Figures 7.8 to 7.12. In Figure 7.8, road tracking started from the upper left corner, with the white line segment showing the location of human input. The following white dots are the road axis points detected by the road tracker. When the texture of the road surface changes, the road tracker failed to predict the next position. Control was returned to the user who entered another road segment as marked by a short line segment. Step prediction scaling strategy enabled the road tracker to work faster. This can be seen in the image, where larger step sizes are used when consecutive predictions were successful.

Figure 7.9 shows how multiple reference profiles help the tracking. Tracking started in the upper left corner. When the road changed from white to black, a match could not be found between the observation and the reference profiles and human input was required as indicated by the white line segment. When the the road changed back to white, no human input was necessary because the profile for white road was already in the list of reference profiles. The tracker searched the whole list for an optimal match.

The performance of the system is influenced by several factors. First, human factors play an important role. Human input is not always accurate; hence the road tracker is affected strongly in the preprocessing step when initial parameters are set and road profiles extracted. This is reflected in similar trends of different filtering algorithms tracking in the same data sets. For example, the efficiency of all trackers is the poorest for user 3 and accuracy is the poorest for user 4. This suggests that our system can also be used to model the inputs given by different users. This, in turn, opens the possibility of investigating user-adapted systems.

Second, the number of particles in particle filter affects the performance of the system,

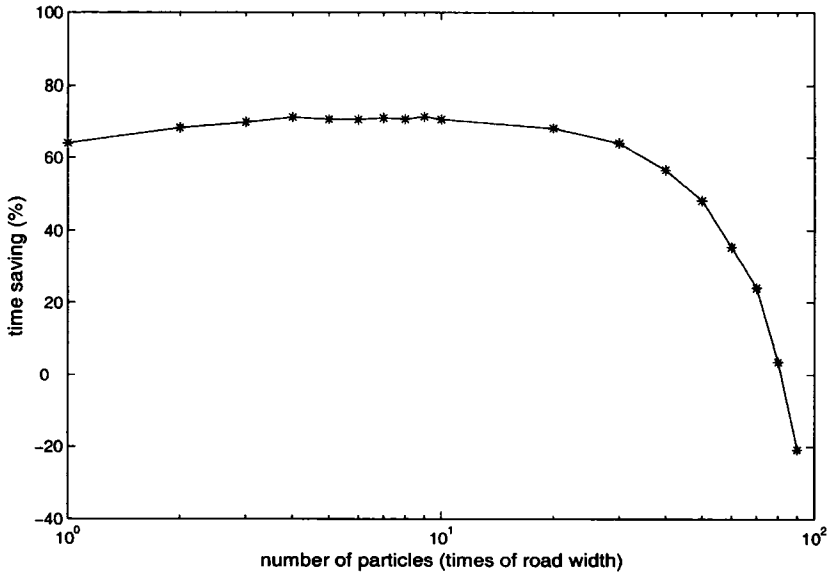


Figure 7.7: The influence of number of particles on the performance of the system on data extracted from user one.

as shown in Figure 7.7. When the number of particles is smaller than 20 times of the road width, the performance of the system is quite steady. However, when this number continues to increase, the performance of the system drops. Though more particles allow for an improved approximation of the posterior density of the state, the system performance decreases due the the time spent on the particle evolution and likelihood computations.

Third, complex road scenes can cause tracking failures, requiring further human input. Figures 7.8 and 7.9 show cases of tracking failures that are caused by abrupt changes of radiometric properties of the road. Figure 7.10 shows a case where tracking stops where road direction changes abruptly. Figures 7.11 and 7.12 show the tracking failures caused by road profile changes due to road connections and occlusions.

When junctions are encountered, the road tracker makes different judgements based on the road condition and the status of the tracker, as shown in Figure 7.13. At junction 1, a matching observation could not be found. But further along the direction of the road, a matching observation profile was found. Thus, junction 1 was jumped over by state updating with a large step size τ due to the jump-over strategy (see Section 8.4) or the step prediction scaling strategy (see Section 7.5.6). However, at junction 2, no matching profile could be found. Thus, the tracking process stopped and control was returned to the human operator. Ultimately, in all difficult situations, it is the human who has to decide how to proceed with tracking.

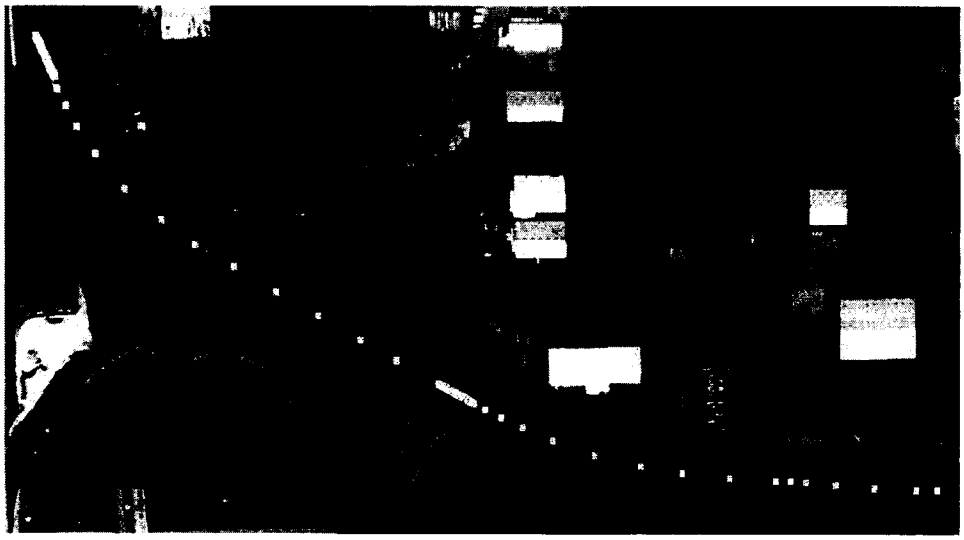


Figure 7.8: Road tracking from upper left to lower right. White dots are the detected road axis points, white line segment shows the location of human input.

7.7 Conclusion

This chapter introduced a human-computer interaction system for robust and efficient road tracking. It attempts to bridge the gap between human and computer in automatic or semi-automatic systems. This approach has a potentially significant impact on the daily work of map revision. It can greatly reduce human effort in the road revision process. At the same time, it guarantees correct and complete results because the user is never removed from the process.

The proposed framework consists of several components, the user, the human-computer interface, computer vision algorithms, knowledge transfer schemes and evaluation criteria. It can compensate for the deficiencies of computer vision systems in performing tasks usually done by humans. This framework also can be applied to systems that require understanding of different levels of interactions between human and computer.

The road tracking method is based on Bayesian filters that match observation profiles to reference profiles. Particle filters and extended Kalman filters are used to predict road axis points by state update equations and to correct the predictions by measurement update equations. During the measurement update process, multiple observations are obtained at a predicted position. The tracker evaluates the tracking result using normalized cross-correlation between road profiles at previous and at the current position. When multiple profiles are obtained from human input, the profile with the highest cross-correlation coefficient is searched, with the most recently used profile being given the highest priority.

Figure 7.9: Road tracking from upper left to lower right. White dots are the detected road axis points, white line segment shows the location of human input. The number of human inputs is reduced by searching multiple reference profile lists, as described in the text.

The use of two-dimensional features, multiple observation and multiple profile methods has greatly improved the robustness of the road tracker. Finally, when they were combined with a step prediction scaling method, tracking efficiency was further increased.

For further progress with making human-machine systems robust and useful, we need to explore a number of paths.

- The simulated system approximated the human-computer interaction in a real-world system. To further study the effectiveness and usability of the system, we need to implement it on an industrial platform, such as in the USGS map revision system.
- The combination of Kalman filters and Particle filters should be studied, especially in developing user-adapted systems.
- Current road tracking model uses the location of road centerline and the direction of the road to make prediction. We could also use higher-order prediction models, which use road curvature or curvature changes in the prediction.
- The system can be more automated and its performance be further improved if more knowledge resources can be involved (for example, the buildings layer).
- The system framework was developed for USGS map revision environment which

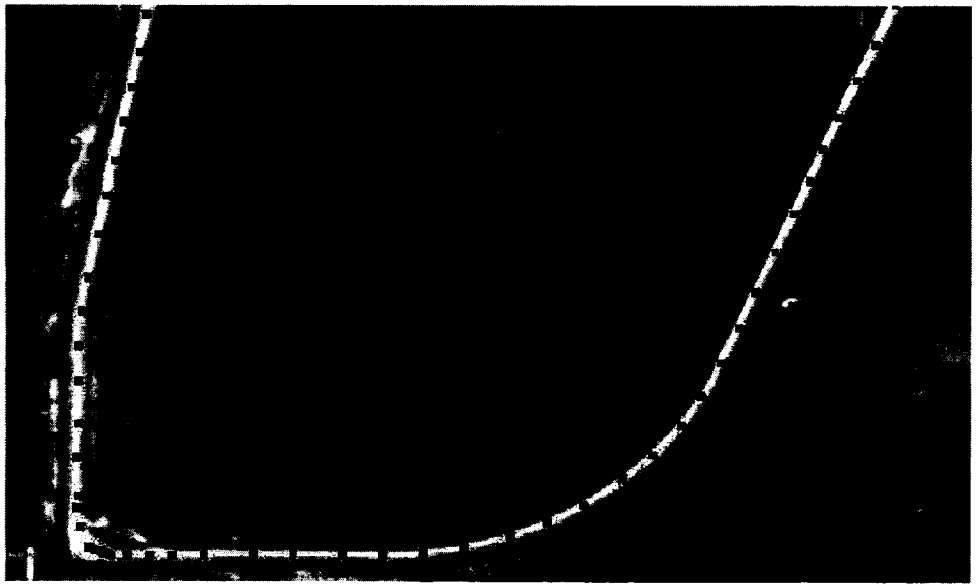


Figure 7.10: Road tracking from upper left to upper right. Black dots are the detected road axis points, black line segment shows the location of human input. The tracking fails when road direction changes dramatically.

uses aerial image as the source of revision. It would be interesting to see how this system can be applied to other types of images, for example, to satellite images.

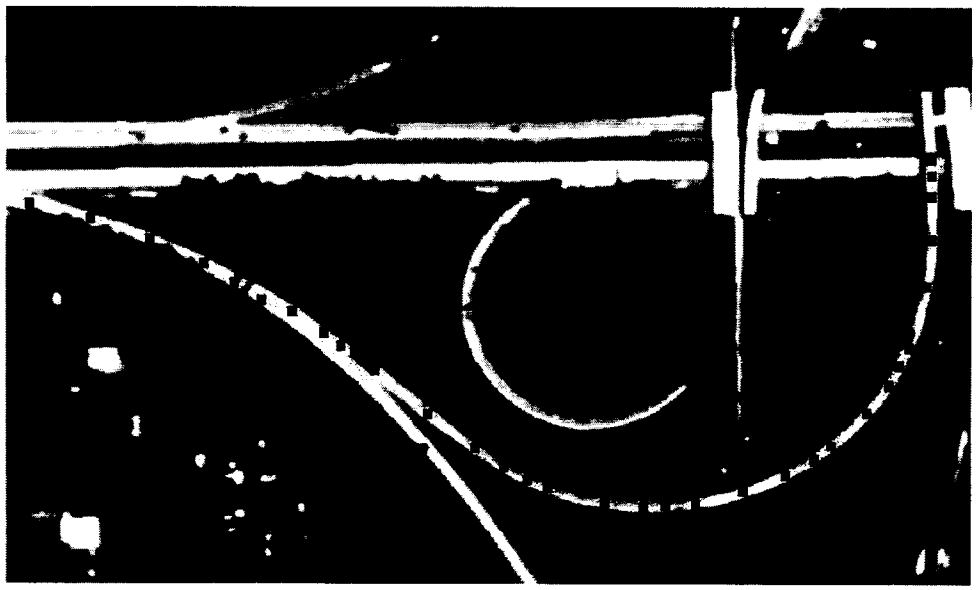


Figure 7.11: Road tracking from upper right to upper left. Black dots are the detected road axis points, black line segment shows the location of human input. The tracking fails when road profile changes at the road connection.

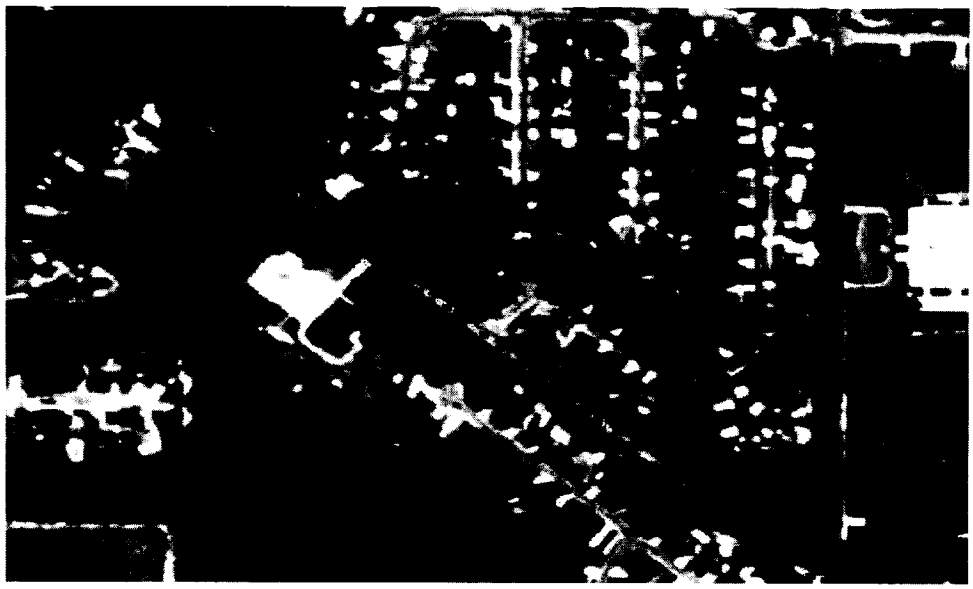


Figure 7.12: Road tracking from upper left to lower right. Black dots are the detected road axis points, black line segment shows the location of human input. This is an extreme case when trees occlude the road. Intensive human inputs are required.

Figure 7.13: Handling of road junctions. Black dots are the detected road axis points. Junction 1 was jumped over, while the tracking stopped at junction 2.

Chapter 8

HCI-based Online Machine Learning for Road Tracking *

8.1 Introduction

We have presented an application of the HCI framework for road tracking in aerial images in Chapter 7. The tracking models use Bayesian filtering and template matching methods to predict the road centerlines. In the template matching, the system compares the observed road profiles with the profiles stored in a knowledge-base, and update this knowledge base with the observations.

In this chapter, we tackle the road tracking problem with another method using the same HCI framework. We introduce an online learning approach [75] that naturally integrates guidance from human experts with automatic computer vision algorithms for tracking roads in aerial photos. Human inputs provide the online learner with training examples to generate road predictors. An ensemble of road predictors is learned incrementally from human inputs, and the predictors are then used to automatically track roads. When novel situations are encountered, control is returned back to the human expert. With this approach, we avoid the problem of having to explicitly define an off-road class, while enabling explicit learning from human inputs. Our learning algorithm is a kernel-based online learning approach for novelty detection. The learned predictors are represented as weighted combinations of training examples, and the weights for each training example are derived from the large margin principle [130]. We discuss the learning methods to acquire single and multiple predictors from one human input, and we show that they are equivalent to learning an expansion model or multiple drifting models that characterize human knowledge of the dynamics of the image patterns. We also introduce two tracking algorithms that use

*A version of this chapter will appear in the Proceedings of the Workshop on Pattern Recognition in Remote Sensing, Hong Kong, China, August 20, 2006.

either optimal candidate-predictor combinations or multiple hypotheses in the prediction. The experimental results of the learning and tracking models are compared and analyzed. The proposed approach is computationally efficient, and it can rapidly adapt to dynamic situations where the road feature distributions change. Experimental results confirm the effectiveness of our approach, and it is shown to be superior to existing methods.

8.2 Related work

8.2.1 Learning from Human-Computer Interaction

Interactive machine learning (IML) [32, 136] enables human operators to train predictors, for example decision trees or neural networks, by providing labelled training data and by defining boundaries between true and false instances. Such learning methods have been successfully used in image segmentation.

Muneesawang and Guan [99] employed IML in a model for content-based image retrieval. An initial image distance function is given using a radial basis function (RBF) network method. When a user supplies a query image, the system retrieves images in the databases that are closest to the query image. Then user points out the correct and incorrect retrievals. The image features are used to train the RBF predictor using learning vector quantization method, so that the system can retrieve a new set of images. This process is repeated until the user is satisfied with the retrieval results.

Maloof et al. [88] proposed a similar model, which enables interactive supervised learning for building detection in aerial image analysis. In each training session, the classifiers generate roof candidates. The human analyst provides positive and negative labels on these candidates, which in turn are used to update the classifiers. The training sessions terminate when all training samples have been classified. Four types of classifiers were implemented, namely a nearest-neighbor method, a naive Bayesian classifier, decision trees, and a continuous perceptron. In the testing session, these classifiers automatically detected buildings in testing data sets and their performance were compared.

The above methods acquire one predictor from human inputs. It is also possible to acquire a set of predictors if multiple predictors are generated from human inputs. In this case, the problem can be considered as predicting from expert advice [14] and it can be solved by online learning algorithms. Littlestone and Warmuth were among the the first to tackle this problem [86]. They proposed a weighted majority algorithm that can be summarized as follows:

1. Set the initial weight w_i for each expert i to 1, where $1 \leq i \leq n$, and n is the number of experts.

2. The algorithm compares the total weight from experts that predict output 0 and the total weight from experts predict output 1, and output the prediction of the larger total.

3. The weights of the experts that made a mistake are multiplied by a penalty factor between 0 and 1.

Kolter and Maloof [78] extended the above algorithm to create and delete experts in concept drift problems using a weighted majority algorithm. In addition to the predictions by each expert, a global prediction is calculated as the label with the highest accumulated weight. Whenever the global prediction is incorrect, the algorithm creates a new expert. To constrain the number of experts, the algorithm eliminates experts with weights less than a certain threshold.

As described in a later section, our online learning problem is similar to the concept drifting problem: each predictor is a weighted sum of past training examples, where the weight for each training example is derived from the large margin principle [130]. Instead of creating new predictors automatically, we maintain a dynamic list of predictors: one set of predictors can be generated from one iteration, and different iterations generate different predictor sets.

We developed our algorithm for novelty detection, *i.e.* to identify new or unknown data that were not present during the training stage [89]. Thus, it is important to find true novel samples in testing while minimizing false positives. Cheng et al. [20] proposed an online learning method with sparse kernels to solve learning problems with examples drawn from non-stationary distribution. We extend this method to online learning from human inputs for novelty detection, so that it can fit into the HCI framework for image interpretation applications.

8.3 One-Class Support Vector Machines

To make this chapter self-contained, we give a brief introduction on one-class Support Vector Machines (1-SVM), which forms the basis of the proposed online learning method. Support vector machines are kernel based classifiers [130]. The purpose of training an SVM is to obtain good separating hyperplanes in a high dimensional space that has optimal generalization ability.

8.3.1 Linear Support Vector Machines

In binary classification problem, given training samples $\{x_i, y_i\} \subseteq \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} \subseteq \mathbb{R}^d$, $\mathcal{Y} = \{-1, 1\}$, and $i = 1, \ldots, l$, a hyperplane in $\mathbb{R}^d$ can be written as

$$\{x \in \mathbb{R}^d | \langle w, x \rangle + b = 0\} \tag{8.1}$$

If the training data is linearly separable, that is, if there exists a hyperplane such that $y_i(\langle w, x \rangle + b > 0)$, we can define two parallel hyperplanes, H_+ and H_-, in the form that

$$H_+ : \min\langle w, x \rangle + b_+ = 0 \tag{8.2}$$

for all positive samples, and

$$H_- : \min\langle w, x \rangle + b_- = 0 \tag{8.3}$$

for all negative samples. Let $b = (b_+ + b_-)/2$ and $\delta = (b_- - b_+)/2$, we can rewrite the above hyperplanes as

$$H_+ : \min\langle \frac{w}{\delta}, x \rangle + \frac{b}{\delta} = 1 \tag{8.4}$$

and

$$H_- : \min\langle \frac{w}{\delta}, x \rangle + \frac{b}{\delta} = -1 \tag{8.5}$$

The separating hyperplane is then transformed into a scaled form

$$\langle \frac{w}{\delta}, x \rangle + \frac{b}{\delta} = 0 \tag{8.6}$$

Given (w, b), suppose all the training data satisfy the following constraints,

$$|\langle w, x_i \rangle + b| \geq 1, \qquad i = 1, \ldots, l \tag{8.7}$$

the margin, which is defined as the distance from the closest samples to the hyperplane $\langle w, x \rangle + b = 0$, is $1/\|w\|$. Then we need to find the pair of hyperplanes that maximize the margin. Introducing Lagrange multipliers $\alpha_i, i = 1, \ldots, l$, one for each of the above inequality constraints, a Lagrangian function can be defined as

$$L_p = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i y_i (\langle w, x_i \rangle + b) + \sum_{i=1}^{l} \alpha_i \tag{8.8}$$

Maximize L_P with respect to w and b vanish,subject to constraints that $\alpha_i > 0$, we have

$$w = \sum_{i=1}^{l} \alpha_i y_i x_i \tag{8.9}$$

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \tag{8.10}$$

Substituting them into Eq. 8.8, we get the dual form

$$L_D(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \tag{8.11}$$

The aim of training support vector machine, is then to maximize L_D, subject to $\alpha \geq 0$ and $\sum_{i=1}^{l} \alpha_i y_i = 0$. Given each α_i solved, the decision function can be written as

$$f(x) = \text{sgn}(\sum_{i=1}^{l} \alpha_i y_i \langle x_i, x \rangle + b) \tag{8.12}$$

where

$$\text{sgn}(u) = \begin{cases} -1 & \text{if } u < 0 \\ 1 & \text{if } u \geq 0 \end{cases} \tag{8.13}$$

8.3.2 Non-linear Support Vector Machines

In order to solve non-linear problems with a linear machine, we need first to map data to another Euclidean space $\mathcal{H}$, in which the mapped data is linearly separable. We also need a way to compute the inner product in the new space directly as a function of the original data. Kernel functions are used for these purposes. Given $x_i, x_j \in \mathcal{X}$, a kernel function k is defined as

$$k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle \tag{8.14}$$

where Φ is a mapping function from the original space to $\mathcal{H}$. The advantage of kernel functions is that we do not need to explicitly calculate the kernel mapping Φ, and we can directly replace $\langle x_i, x_j \rangle$ with $k(x_i, x_j)$ in the training process.

An important concept associated with the kernel mapping is Reproducing Kernel Hilbert Space (RKHS). Suppose we have the following reproducing kernel mapping

$$\Phi : x \longrightarrow k(\cdot, x) \tag{8.15}$$

Then we can construct a RKHS as

$$f(\cdot) = \sum_{i=1}^{l} \alpha_i k(\cdot, x_i) \tag{8.16}$$

which is a vector space of all linear combination of the kernel functions $k(\cdot, x)$. This RKHS has the important properties $f(x) = \langle k(\cdot, x), f(\cdot) \rangle$ and $\langle k(\cdot, x_i), k(\cdot, x_j) \rangle = k(x_i, x_j)$.

8.3.3 One-class Support Vector Machines

When only data in one class is available, SVM must be modified to detect outliers. Schölkopf et al. proposed an algorithm to solve this problem by estimating a hyperplane based on data that lie on one side of the boundary [119], which is called one-class support vector machines (1-SVMs).

The strategy of the algorithm is to first map the data into a feature space using a kernel function, then to set the origin as the only sample in the second class and to separate data from the origin with maximum margin. To do so, equation 8.8 becomes

$$L_p = \frac{1}{2}\|w\|^2 - \frac{1}{vl}\sum_{i=1}^{l}\xi_i - b - \sum_{i}^{l}\alpha_i((\langle w, \Phi(x_i)) - b + \xi_i) - \sum_{i=1}^{l}\beta_i\xi_i \tag{8.17}$$

where $v \in (0,1)$ is a bound that a training point is outside of the estimated region, ξ_i is a nonzero slack variable, α_i and β_i, $i = 1, \ldots, l$, are Lagrange multipliers. Maximizing L_P, subject to $< w, \Phi(x_i) >\geq b - \xi_i$ and $\xi_i \geq 0$, we have

$$w = \sum_{i}^{l}\alpha_i(\Phi(x_i)) \tag{8.18}$$

$$\alpha_i = \frac{1}{vl} - \beta_i \leq \frac{1}{vl}, \quad \sum_{i=1}^{l} = 1 \tag{8.19}$$

Given the solution for each α_i, the decision function can be written as

$$f(x) = \text{sgn}(\sum_{i=1}^{l}\alpha_i k(\langle x_i, x\rangle) - b) \tag{8.20}$$

where

$$\text{sgn}(u) = \begin{cases} -1 & \text{if } u < 0 \\ 1 & \text{if } u \geq 0 \end{cases} \tag{8.21}$$

When $f(x) = -1$, x is classified as an outlier.

8.4 System framework

In this section, we give an overview of the main procedures of the proposed approach. In contrast to Chapter 7, the human-computer interaction process is redefined to fit the machine learning framework.

A road tracking task starts with an aerial photo with the target of tracking, and the system proceeds with iterations consisting of human input, sampling and prediction phases. The first phase is aimed at learning from human input whereas the last phases are aimed at automatic road tracking.

Input phase. A human input consists of two mouse clicks on an image, with the line joining the click positions defining a road axis and indicating the road direction. Along the road direction, a set of road profiles are extracted normal to and along the road axis at consecutive axis points. The length of each profile is determined by the road width, which is estimated from the distance between the road edges. These, in turn, are obtained with a gradient-based edge detection method [148]. We denote a road profile as $x \in \mathcal{X} \subseteq \mathbb{R}^d$ where $\mathcal{X}$ is the profile space with dimension d, and we denote one human input as one learning session $s \in \mathcal{I}_S$ where $\mathcal{I}_S$ indexes the set of learning sessions that occurred for this road tracking task. A road profile x is associated with a label $y \in \mathcal{Y}$, where $\mathcal{Y}$ is the set of feasible labels (e.g. $y = 1$: on the road, $y = 0$: off-road), and a state $\sigma \in \Sigma$, which encodes a location as

$$\sigma = \begin{bmatrix} u & v & \theta \end{bmatrix}', \tag{8.22}$$

where u and v are the coordinates of road axis point, and θ is the direction of the road. The triplet $z = (x, \sigma, y)$ represents an *example*. We further restrict that, at an appropriate road axis location, there has to be exactly one profile with $y = 1$, hence $y = 1$ for all training examples. To keep the notation simple, we assume that there are T road profiles along the entire road, and we define a set of time steps $\mathcal{T} = \{1, \cdots, T\}$ with one road profile for each time step. As will become clear later, a predictor (*i.e.* a road profile predictor) $f_s \in \mathcal{F}$ is learned from a learning session s, where $\mathcal{F}$ denotes the set of predictors learned in sessions indexed by $\mathcal{I}_S$.

Sampling phase. We assume that, at time t, the system has experienced a sequence of learning sessions $(1, \cdots, s)$ and obtained an ensemble of predictors $\mathcal{F}_1^s \triangleq (f_1, \cdots, f_s)$ by $\mathcal{F}_1^s = \mathcal{F}_1^{s-1} \cup f_s$. It then proceeds with the sampling phase, where the system searches the neighborhood (bounded by a fixed range of angles and depths) along the current road axis for candidate road profiles $\tilde{\mathcal{X}} = \{x_1, \cdots, x_m\}$, where m denotes the number of candidates being sampled. The state at time t is sampled using the following non-linear function

$$\sigma_t = \begin{bmatrix} u_{t-1} + \varrho \cos(\theta_{t-1}) \\ v_{t-1} + \varrho \sin(\theta_{t-1}) \\ \theta_{t-1} \end{bmatrix} \tag{8.23}$$

where ϱ is the step size of the sampling determined by the road width and tracking status.

Prediction phase. After the sampling phase, the prediction phase starts with a set of candidate profiles $\tilde{\mathcal{X}}$ and their associated states, with the goal of picking a predicted profile $\hat{x}_t$ and predicting its label $\hat{y}_t$. If $\hat{y}_t = 0$ (the profile is not on the road, *i.e.* a novelty), control is handed back to the human expert for further input. In this manner, switching between

human inputs and automatic tracking continues until the tracking task is completed.

For predicting an example $\hat{z}_t = (\hat{x}_t, \hat{\sigma}_t, \hat{y}_t)$, we hope to be close to the true one $z_t = (x_t, \sigma_t, y_t)$, that is, $\hat{x}_t \rightarrow x_t$, $\hat{\sigma}_t \rightarrow \sigma_t$ and $\hat{y}_t = y_t = 1$. Unfortunately, this is not always the case. When cars or long trucks are presented on the road, none of the candidate profiles may match. As introduced in Section , a heuristic strategy was developed to overcome these often-encountered situations, where the step size ϱ in Equation 8.23 is increased to jump over the current state when the system fails to find a road profile with $\hat{y} = 1$. Then a new sampling phase occurs from the previous state. When failures continue, even with the jump-over strategy, the system recognizes a tracking failure and returns control to the human expert, who then inputs another road segment from which new training examples with $y = 1$ can be extracted.

In summary, the learning approach is naturally decomposed into two parts, a learning algorithm devised for the human input phase, and a tracking algorithm for automatic road tracking.

8.5 Learning Algorithms

The interactions between human and computer lead to a situation, where learning sessions are mixed with automatic tracking runs. The first learning session is initialized by the first human input. Each successive learning session s starts when, at time t, an outlier is detected ($\hat{y}_t = 0$) for the current predicted profile $\hat{x}_t$ and control is handed back to the human expert. For the sake of simplicity, we assume that each learning session contains exactly S examples. Therefore, the learning session finishes when the expert finishes teaching, with inputs consisting of successive examples $(z_{t+1}, \cdots, z_{t+S})$, where $z_i = (x_i, \sigma_i, y_i), \forall i \in \{t+1, \cdots, t+S\}$.

8.5.1 Basic learning algorithm

One learning session corresponds to one human input, with the goal of obtaining a reasonable predictor f_s. Assume that the current learning session s starts at time t and contains exactly S examples $(z_{t+1}, \cdots, z_{t+S})$. Further, we define a kernel mapping $k(\cdot, \cdot)$ from profile space to a Hilbert feature space, $\mathcal{X} \rightarrow \mathcal{H}$ as $x \mapsto k(\cdot, x) \in \mathcal{H}$. Here $\mathcal{H}$ denotes the Reproducing Kernel Hilbert space (RKHS) with induced kernel $k(\cdot, \cdot)$ such that $f(x) = \langle k(\cdot, x), f(\cdot) \rangle$, and $\langle \cdot, \cdot \rangle$ gives the inner product. The norm in this case is naturally defined as $\| \cdot \| = \langle \cdot, \cdot \rangle^{1/2}$.

As in the online learning algorithm described in [20], the predictor $f \in \mathcal{H}$ can be represented as a weighted combination of training profiles, where past examples in the learning session $\{z_i\}_{i=t+1}^{t+S}$ are associated with different weights $\{\alpha_i\}_{i=t+1}^{t+S}$ that are derived formally from the large margin principle [130]. We extend this algorithm to incorporate learning from human inputs and to deal with the novelty detection scenario, so that the learning problem is naturally formulated as a novelty detection by solving online 1-SVMs.

Given a profile x_i at time i, the novelty detection can be formulated as a linear program in a manner similar to 1-SVMs introduced in Section 8.3.3

$$\begin{aligned} &\min_{\|f\|=1,\xi} \quad C\xi \\ &s.t. \quad \langle f, k(x_i, \cdot)\rangle \geq \gamma - \xi, \\ &\qquad\qquad \xi \geq 0 \end{aligned} \tag{8.24}$$

where γ, C are positive constants, and ξ is a positive slack variable. We also define the loss function as

$$l_i \triangleq l(f_i; x_i) = (\gamma - (1-\tau)\langle f_i, k(x_i, \cdot)\rangle)_+ \tag{8.25}$$

where $(\cdot)_+ \triangleq \max\{\cdot, 0\}$. According to the framework in [20], we want to minimize the risk function

$$R = R_{\text{div}}(f) + R_{\text{reg}}(f) \tag{8.26}$$

Here,

$$R_{\text{div}}(f) = \|f - f_i\|^2/2 \tag{8.27}$$

measures the distance of the predicted f from the previous prediction f_i. The second term is

$$R_{\text{reg}}(f) = \underbrace{\frac{\lambda}{2}\|f\|^2}_{\lambda R_{\text{cap}}(f)} + \underbrace{C\xi + \varsigma(\gamma - \langle f, k(\cdot, x_i)\rangle - \xi) - \zeta\xi}_{R_{\text{inst}}(f)}, \tag{8.28}$$

where ς and ζ are Lagrangian multipliers, and $\lambda \geq 0$ is a regularization parameter. Equation (8.28) consists of two terms, the capacity risk, $R_{\text{cap}}(f)$, which controls the complexity of the prediction f; and the instantaneous risk, $R_{\text{inst}}(f)$, which is the Lagrangian function of the optimization problem in Equation (8.25). Further, we introduce $\tau, \upsilon \geq 0$, such that $\lambda = \tau/(1-\tau)$ and $\varsigma = \upsilon/(1-\tau)$. By solving the (previously mentioned) risk minimization problem, as shown in [20], the separating function f turns out to be $f(\cdot) = \sum_{i=t+1}^{t+S} \alpha_i k(\cdot, x_i)$, where the weights are

$$\begin{cases} \alpha_j & \Leftarrow (1-\tau)\alpha_j \quad \forall j < i \\ \alpha_i & \Leftarrow \min\left\{\frac{l_i}{k(x_i,x_i)},\ (1-\tau)C\right\} \end{cases} \tag{8.29}$$

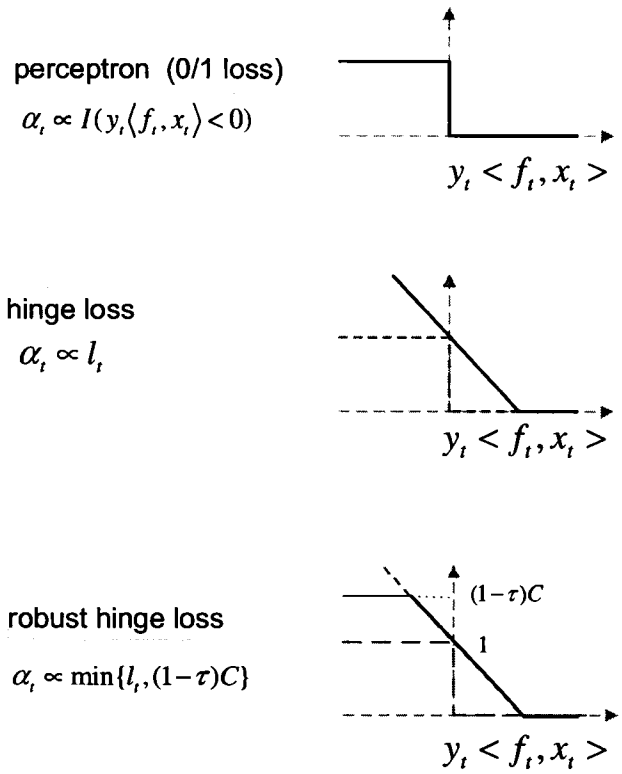


Figure 8.1: Robust hinge loss function for α_i assignment at time t. α_i is upper bounded by $(1-\tau)C$.

As stated in [20], the resultant weight updating formula has several advantages. First, the robust hinge loss ensures limited influence from outliers (see Figure 8.1). In the perceptron, $0/1$ loss leads to same weight assignment to all outliers. In regular hinge loss, weights are assigned to outliers proportional to the incurred loss. It means that some outliers may have much higher influence to the solution than the others. Both cases are not desired. Using the robust hinge loss, we can ensure that α_i is always upper bounded by $(1-\tau)C$. Second, by adjusting the decay rate $\tau \in [0, 1)$ to an appropriate value, the new predictor f is able to balance between two extreme situations, either fully adapting to the current example (*i.e.* forgetting all the past examples as $\tau \to 1$), or keeping all past examples in memory (*i.e.* becoming a batch learning case as $\tau \to 0$). Finally, at time i, the weight of current example x_i, α_i is automatically obtained, while the weights for previous examples $\forall j < i$ are retained with proper decay, instead of being re-computed from scratch. Thus, this update formula allows for rather efficient computation, as shown in the learning algorithm below.

Based on the Learning Algorithm, the predictor f_s is obtained given the weight sequence $(\alpha_i^{(s)})_{i=t+1}^{t+S}$ and the corresponding training examples. For a sequence of length S,

Algorithm 8 Online learning algorithm

Learning Algorithm

Input: The cut-off value C, decay rate τ, current learning session s.

Initialize: $f_{t+1} \Leftarrow \underline{0}$.

  for $i = t+1$ to $t+S$ **do**

    Observe profile x_i

    Compute $l_i^{(s)}$ according to Equation (8.25)

    Compute $(\alpha_j^{(s)})_{j=t+1}^{i}$ according to Equation (8.29)

  end for

Output: The sequences $(\alpha_j^{(s)})_{j=t+1}^{t+S}$, s, f_s.

the space complexity of the proposed learning algorithm is $(d+1)S$, and the time complexity is $O(S^2)$.

8.5.2 Learning multiple predictors from one training session

We can generate multiple predictors from one training session. Since the separating function is in the form of $f(\cdot) = \sum_{i=t+1}^{t+S} \alpha_i k(\cdot, x_i)$, given an input that consists of successive examples $(z_{t+1}, \cdots, z_{t+S})$, we can compute up to S separating functions from $t+1$ to $t+S$. Thus, an ensemble of predictors is given in the form of $\mathcal{F}_1^s \triangleq (f_{1,1}, f_{1,2}, \cdots, f_{i,j}, \cdots, f_{s,S-1}, f_{s,S})$. For a sequence of length S, the time complexity of this extended model is not different from the single-predictor model, but the space complexity increases to $(d+1)S^2$.

In each training session, some predictors may not be as robust as others. To evaluate their robustness, we calculate the training errors for all predictors and rank them according to their performance. This permits us to select more robust predictors in the tracking runs and to achieve faster and more accurate results.

8.5.3 Discussion of the learning algorithms

As mentioned previously, the online learning algorithm is derived from 1-SVMs [119], and it is thus driven primarily by positive examples. This formulation perfectly fits the road tracking scenario, since the human inputs provide only positive examples for the learning process. In the following, we analyse the effect of applying this algorithm to a set of 2D data as illustrated in Figure 8.2. We assume a Gaussian kernel, and the learning algorithm generates, in each learning session, a sphere enclosing the features from input data.

We consider two learning models. In one model, we use a decay rate $\tau = 0$, where all past examples contribute equally to a predictor without any loss of information. Learning thus expands the region to cover the new example. In the model with decay rate $\tau \neq 0$,

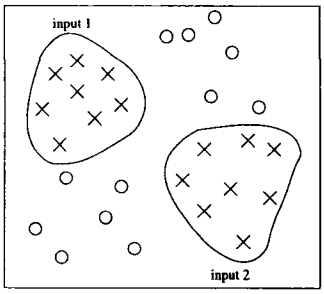


Figure 8.2: This graph shows a set of input data. Initially, the class of the data is unknown. A human input sequentially mark the data that human operator considers as positive, which are in turn used in one training session. The region delimited by one curve identify these positive examples. In the proposed interactive model, multiple inputs can be observed, and thus form multiple regions as marked by input 1 and 2.

the region drifts to adapt to the new example, and the extent of the drifting depends on the value of τ. With a large τ, drifting is larger and past examples are forgotten more quickly (see Figure 8.3).

Since old examples characterize situations that are considered positive, we may want to preserve the past training examples in the drifting model. In this case, multiple predictors can be learned from one human input. The predictors acquired in each time step are preserved, each becoming an independent expert for the tracking process. This model is shown in Figure 8.4.

8.6 Tracking and Novelty Detection Algorithms

The tracking process automatically interprets road features in the aerial images using the trained predictors. If a novelty is found in the direction of the prediction, then either a new road condition has been observed or a tracking failure has been encountered. In both cases, human guidance is required to initialize a new training session, and if necessary, to correct the tracking error. We now introduce two tracking algorithms, one based on a single optimal predictor, the other using multiple predictors.

8.6.1 Optimal Candidate-Predictor Combination

We assume that, at time t, after the *sth* learning session, the newly learned predictor f_s is incorporated into the set of predictors as $\mathcal{F}_1^s = \mathcal{F}_1^{s-1} \cup f_s$. Tracking starts by searching the neighborhood along the current road axis for candidate profiles. The losses of the candidates are calculated using Equation (8.25) and the one with the minimum loss, $\hat{x}_t$, is picked as

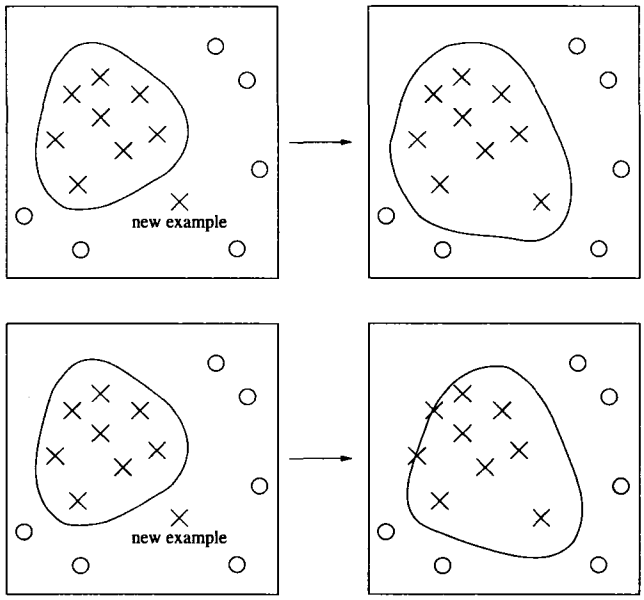


Figure 8.3: Two models of learning: expansion (upper graph) and drifting (lower graph). The left graphs shows the learning result up to time t. At time $t+1$, a new positive example is observed. The expansion model enlarges the region delimited by a curve to include the new example, while the drifting model moves the region to adapt to the new data.

the input to the predictors. If it is considered to be on the road ($\hat{y}_t = 1$ with the predictor $f \in \mathcal{F}$, which produces the least loss), the state $\hat{\sigma}_t$ of the profile is used to set the current road axis point and the origin of the next prediction. The tracking algorithm based on optimal candidate-predictor combination is given in Algorithm 9.

Algorithm 9 Tracking Algorithm I: Optimal Candidate-Predictor Combination

Tracking Algorithm I

Input: Decay rate τ, threshold ϵ, the set of learned predictors so far $\mathcal{F}_1^s$.

Obtain a set of m candidate profiles $\tilde{\mathcal{X}}$ from the sampling phase.

for $i = 1$ to m, $j = 1$ to s **do**

Compute $l_{i,j}$ according to Equation (8.25)

end for

$(l^*, \hat{x}_t) \leftarrow \min_{i,j}\{l_{i,j}\}$

Predict label as

$$\hat{y}_t = \begin{cases} 0, & l^* \geq \epsilon \\ 1, & otherwise \end{cases} \tag{8.30}$$

Output: $\hat{z}_t = (\hat{x}_t, \hat{\sigma}_t, \hat{y}_t)$

The tracking algorithm picks the candidate-predictor combination with minimum loss, and it evaluates whether the candidate is on the road and its reliability by comparing the loss to a threshold.

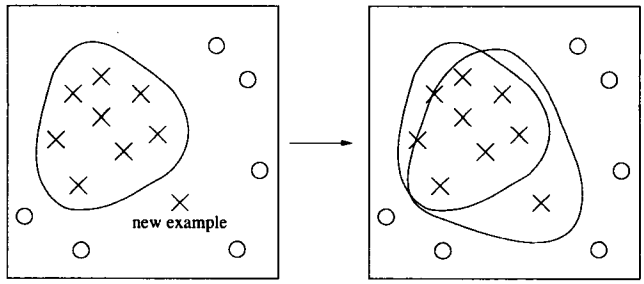


Figure 8.4: Learning multiple predictors from one human input. The learned model at time t is shown in the left graph. At time $t+1$, a new model is learned to adapted to the new training example, while the old model is kept.

8.6.2 Multiple-Hypotheses Support

The road axis point and next prediction depend exclusively on the location and direction of the optimal candidate. Noisy candidates can thus mislead the prediction. To solve this problem, we propose a multi-hypothesis method. The losses of the candidates are again calculated using Equation (8.25) for each $f \in \mathcal{F}$. These losses are stored in a look-up table $\mathcal{H}$, whose entries $h_{i,j} \in \mathcal{H}$ correspond to hypothesis road axis points from candidate-predictor combinations.

Given the lookup table, we search for hypotheses whose losses are lower than a threshold ϵ. If a loss $l_{i,j}$ is smaller than ϵ, the hypothesis is a supporting hypothesis and its weight is set to

$$w_{i,j} = \exp(-l_{i,j}) \tag{8.31}$$

If $l_{i,j} \geq \epsilon$ the hypothesis is discarded. Finally, we calculate the road axis point as a normalized weighted sum of the supporting hypotheses. If the number of supporting hypotheses is zero, a novelty is detected. The resulting tracking algorithm is given in Algorithm 10.

8.7 Experimental Results

We performed experiments on the same platform as introduced in Chapter 7. The same applies to the user data collected and the evaluation criteria.

8.7.1 Experimental results

We compared the results of the proposed Online Learning and Novelty Detection (OLND) algorithm with the two algorithms reported in Chapter 7. In those two tracking algorithms, observed profiles were matched with reference profiles using cross correlation. The state

Algorithm 10 Tracking Algorithm II: Multiple-Hypotheses Support

Tracking Algorithm II

Input: Decay rate τ, threshold ϵ, the set of learned predictors so far $\mathcal{F}_1^s$, an empty $\mathcal{H}$.

Obtain a set of m candidate profiles $\tilde{\mathcal{X}}$ from the sampling phase.

for $i = 1$ to m, $j = 1$ to s **do**

Compute $l_{i,j}$ according to Equation (8.25)

$h_{i,j} \leftarrow l_{i,j}$

$\mathcal{H} = \mathcal{H} \cup h_{i,j}$

end for

for $i = 1$ to m, $j = 1$ to s **do**

if $l_{i,j} < \epsilon$ **then**

compute $w_{i,j}$ according to Equation (8.31)

else

$\mathcal{H} = \mathcal{H} - h_{i,j}$

end if

end for

$w_{i,j} = w_{i,j} / \sum_i \sum_j w_{i,j}$

$\hat{x}_t \leftarrow \sum_i \sum_j w_{i,j} x_i$

$\hat{\sigma}_t \leftarrow \sum_i \sum_j w_{i,j} \sigma_i$

Predict label as

$$\hat{y}_t = \begin{cases} 0, & \mathcal{H} = \varnothing \\ 1, & otherwise \end{cases} \tag{8.32}$$

Output: $\hat{z}_t = (\hat{x}_t, \hat{\sigma}_t, \hat{y}_t)$

prediction was implemented with Kalman filtering (CCKF) and particle filtering (CCPF). The OLND algorithm learned one predictor from each human input and performed novelty detection using the optimal candidate-predictor combination model.

A comparison of results obtained with the three algorithms CCKF, CCPF and OLND is given in Table 8.1. The results show a clear but statistically not significant trend, namely an improvement of the tracking performance using the proposed OLND algorithm compared to the CCKF and CCPF algorithms. OLND algorithm achieves a higher degree of automation and efficiency, as well as better accuracy. A more detailed comparison of results is shown in Figure 8.5, which indicates an improved performance of the OLND algorithms for most user data sets.

To observe how the two learning choices affect the overall performance of the semi-automatic image interpretation, we performed experiments on learning models with different decay rates. Figure 8.6 shows the results for one user. We found that the system achieves the highest tracking efficiency with a decay rate of 0.2, but this occurs at the cost of a reduced tracking accuracy. It means that an appropriate drifting model is more robust and efficient as it balances the old and recent training examples, which better characterize

Table 8.1: Comparison of road tracking results. The meaning for algorithms and comparison criteria are described in the text.

	input saving (%)	distance saving (%)	time saving (%)	RMSE (in pixels)
CCKF	71.9	85.3	63.9	1.86
CCPF	72.3	85.6	62.0	1.90
OLND	75.0	87.8	69.1	1.54

Table 8.2: Comparison of tracking algorithms: TAI for the optimal observation-predictor combination model, and TAII for the multiple-hypotheses support model. The parameters for the model are: $\tau = 0.05, \gamma = 1, \epsilon = 0.99$

	input saving (%)	distance saving (%)	time saving (%)	RMSE (in pixels)
TAI	75.2	87.7	68.3	1.44
TAII	76.2	87.8	71.2	1.62

the gradual changes of the road feature.

Figure 8.7 shows the results of acquiring different number of predictors from one human input. Notice that the purpose of prediction is to find the true road axis, hence tracking continues as long as a road axis can be found. As more predictors are learned from human inputs, the probability of getting at least one prediction on the road increases, allowing the automatic tracking to last longer. On the other hand, noise present on the road surface may affect the predictors, which in turn may generate false positive or false negative novelties in the tracking phase. An increase in the number of predictors also increases the probability that noise affects the model. This leads to a trade-off in the tracking results: the robustness of the model rises first, then drops, when the number of predictors increases. Another negative effects of more predictors is that the tracking efficiency decreases.

Finally, we compare the performance of two tracking algorithms, in Table 8.2 and Figure 8.8. In both experiments, a single predictor was acquired in the training session from each human input. The results show a clear, but statistically not significant trend, namely that the multiple-hypotheses support model is better than the optimal candidate-predictor combination model, both with respect to robustness and efficiency. The result suggests that multiple-hypotheses helps the automatic process. When noise is present on the road, the optimal candidate-predictor combination is likely to deviate from the true road axis point. With the weighted sum of multiple hypotheses, this deviation effect is reduced.

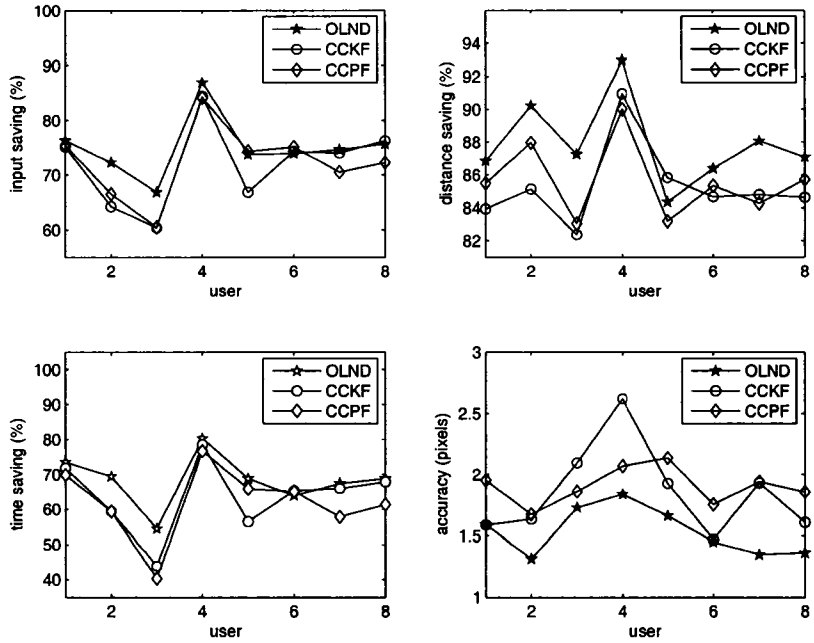


Figure 8.5: Comparison of tracking performance for the proposed algorithm (OLND) and the cross correlation with Bayesian filtering algorithms (CCKF and CCPF). The horizontal axis shows data sets recorded from different users. The vertical axis shows different evaluation criteria. The parameters for the OLND model are: $\tau = 0$, $\gamma = 1$, $\epsilon = 0.95$. For display purposes, the data points are connected by lines even though there is no quantitative relationship between data sets.

8.8 Conclusion

We have presented an online learning approach for novelty detection in image feature tracking. This approach is applied to road tracking in aerial images within a human-computer interaction framework that enables natural switching between human inputs and automatic tracking. It fills the gap between human and computer in image interpretation applications.

Depending on requirements, single or multiple predictors can be learned from the human inputs. These predictors then automatically track road using either an optimal candidate-predictor combination model or a multiple-hypotheses support model. We analyzed the theoretical and experimental difference between the learning and tracking models. Results show that multiple predictors from one human input further helps the learning. However, the system performance drops when too many predictors are acquired. In case of tracking models, the multiple-hypotheses support model outperforms the optimal candidate-predictor combination model.

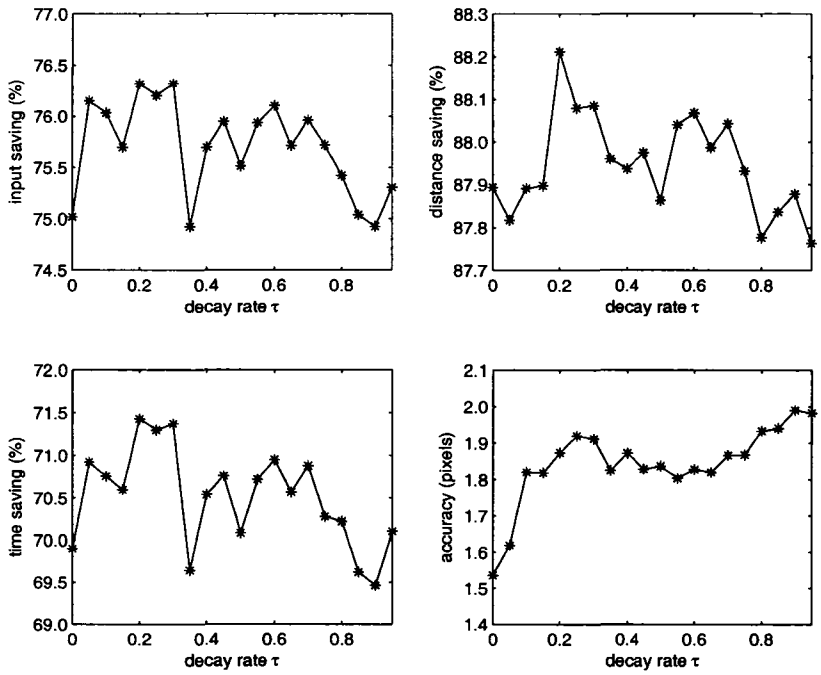


Figure 8.6: Comparison of tracking performance with different decay rates in the learning session. The parameters for the model are: $\gamma = 1, \epsilon = 0.95$

Besides conceptual advantages, the experiments on real world tasks validated the superior performance of the proposed approach, compared to models without machine learning. Our approach is very generic and could be applied to similar applications that require intensive human-computer interactions.

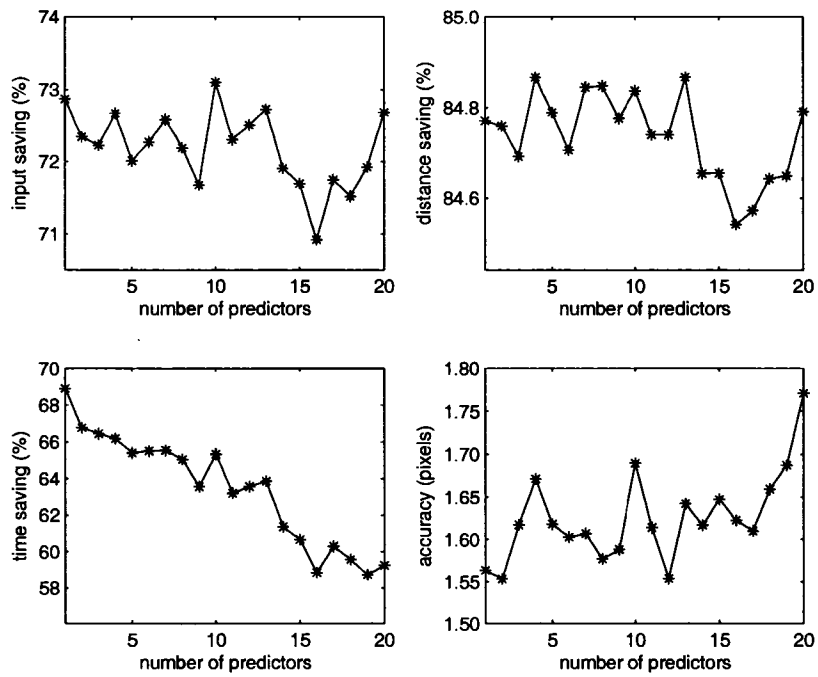


Figure 8.7: Effects of acquiring multiple predictors from one human input. The parameters for the model are: $\tau = 0.2$, $\gamma = 1$, $\epsilon = 0.95$.

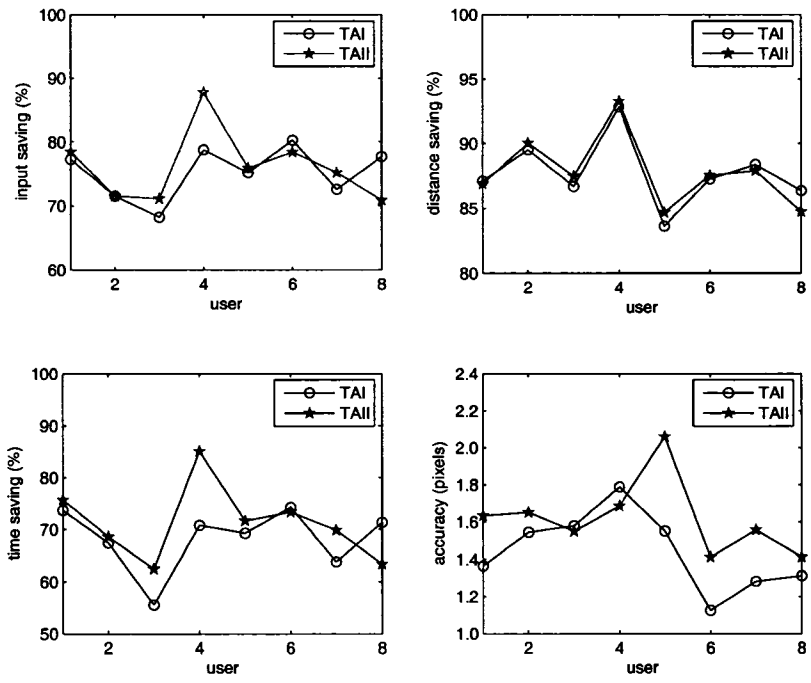


Figure 8.8: Comparison of the tracking algorithms: TAI for the optimal observation-predictor combination model, and TAII for the multiple-hypotheses support model.

Chapter 9

Conclusions

9.1 Summary of the Thesis

In this thesis, we proposed a human-computer interaction framework for image interpretation systems. It is an attempt to fill the gap between human and computer in automatic or semi-automatic image interpretation. This framework consists of a number of components, human-computer interface, user modelling, image interpretation models, knowledge transfer schemes and evaluation criteria.

We introduced an application in semi-automatic map revision, and we reported on two experiments, one on predicting human viewing changes, and the other comparing the performance of human and computer in road tracking. We concluded 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.

Applying the proposed framework to remote sensing image interpretation, we developed two methods for robust and efficient road tracking from aerial images. The computer tracks all human actions and learns a task by modelling human action and comparing the human actions with the image input. On request, the computer takes over simple tasks, returning control to human as soon as confidence rating gets too low. This approach has potentially significant impact on the daily work of map revision because it can greatly reduce the human effort in the road revision process, while guaranteeing accurate results.

The first reported road tracking method is based on Bayesian filters that use a multiple observation profile matching model. The road tracker first estimates the road width and extracts an initial road profile from the human input using edge detection. Then a Bayesian filter is used to predict road axis points by state update equation and correct the predictions by measurement update equation. During the measurement update process, multiple observations are obtained at the predicted position. The tracker evaluates the tracking result

using normalized cross-correlation between road profiles at previous and at current positions. When multiple profiles are obtained from human input, the profile with the highest cross-correlation coefficient is searched, with the most recently used profile being given the highest priority in the search. From time to time, the tracker fails to find points with a sufficiently high cross-correlation. These points are skipped, and control is returned to the human operator if too many points are skipped. The use of two-dimensional features, multiple observations, and multiple profiles has greatly improved the robustness of the road tracker. When they were combined with multiple scale methods, tracking efficiency was further increased.

The second road tracking method uses the same observation model as the first one. However, an online learning and prediction for novelty detection model was proposed and implemented for the tracking process. The tracking switches between human and computer. Human inputs are used to train a set of road profile predictors. From one human input, either single or multiple predictors can be learned. These predictors then track the road centerline automatically. Two prediction models were presented, an optimal candidate-predictor combination model and a multiple-hypothesis support model. Both the learning and predicting models were analyzed with respect to theoretical and experimental aspects. Results show that multiple predictors from one human input further helps the learning. However, the system performance drops when too many predictors are acquired. In case of tracking models, the multiple-hypotheses support model outperforms the optimal candidate-predictor combination model.

The performance of the two road tracking methods was compared in degree of automation, efficiency and accuracy. In the first model, particle filters and extended Kalman filters were implemented. In the second model, a single predictor was obtained in the training step, and optimal candidate-predictor combinations were used in the prediction. Experimental results validated the superior performance of the online learning and prediction model. It suggests that learning from human is necessary in order to achieve a better road tracking system.

9.2 Future Work

This work has done some initial work in HCI for image interpretation. As an important and practical scheme, it is important to extend the theory and see how its working in real systems.

First, we need to further study the learning models. In this thesis, bootstrap learning technique has been used to model the human-computer interactions, in which multiple predictors are learned through the interactions. To reduce the computational complexity, one predictor can be learned to perform the road tracking. Some machine learning methods can fit in the problem very well, such as boosting [38] and reinforcement learning [125]. The advantage of boosting is that it can combine several weak learners into an arbitrarily strong learner. In our case, at the beginning of the interactions, the predictors are weak. Thus, boosting is expected to generate a more robust predictor. In using the reinforcement learning to solve the problem, the human can be considered as part of the environment, which interacts with a goal-seeking agent, the road predictor.

Second, new tracking models should be incorporated into the proposed system. The particle filter used in the system is a sampling importance resampling (SIR) filter. The importance sampling density for the SIR filter is independent of measurement, thus the most recent observation is missing. Further, the resampling in each iteration may cause rapid loss of diversity in particles. The particles may quickly collapse into the same point in the state space [4]. It is known that regularized particle filters (RPF) and unscented particle filters (UPF) are good solutions to these problems [4, 116]. We will look into these models and their application in our system.

Third, in order to make the system more practical, we have to consider more usability issues in designing the interaction framework. This includes using and developing new interfaces for the human input, both in hardware and in software, for example, pupil tracking devices, electroencephalographic devices, or GUIs, so that more user information can be capture and analyzed. It also includes user modelling in multi-task systems in order to build more complex prediction models on human intention, for example, for modelling the sequence of the steps that a user performs, and reason about the status that a user is in. Then we know more timely and precisely when and how to provide help to the human. Current control of the system depends on the internal evaluation by the computational model. When the automatic system is not confident enough to its performance, it stops and hands over control to the human. More usability issues should be considered for better control mechanism, for example, whether and how human can terminate an automatic process when things go wrong, and how to develop more robust internal evaluation methods.

Fourth, we need to consider the quality control problem in the human input because it greatly affects the performance of the HCI system. We hope to avoid situations where a new user enters wrong or misleading inputs to the system. One solution is to build expert

diagnosis models that incorporate knowledge from human experts and existing correct image interpretation results. Another solution is to build training systems with specific image interpretation tasks for new users, so that they can get familiar with what to do and how to do correctly, before they start the real work.

Last, it is interesting to apply the interaction framework to other tasks in the map revision application, such as building revision, boundary revision, and task combination. This requires that we integrate the change detection and production into a coupled task, in which human inputs not only initialize the production step, but also give hints to the automatic change detection model.

Bibliography

[1] R. Adams and L. Bischof. Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):641–647, 1994.

[2] J. Arnqvist, L. Hellgren, and J. Vincent. Semiautomatic classification of secondary healing ulcers in multispectral images. In *Proceedings of the 9th International Conference on Pattern Recognition*, volume 1, pages 459–461, 1988.

[3] A. Arsenio. Map building from human-computer interactions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, volume 10, pages 151–157, Washington DC, June 2004.

[4] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filter for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

[5] R. Bajcsy and M. Tavakoli. Computer recognition of roads from satellite pictures. *IEEE Transactions on Systems, Man, and Cybernetics*, 6(9):623–637, 1976.

[6] E. Baltsavias. Object extraction and revision by image analysis using existing geodata and knowledge: current status and steps towards operational systems. *ISPRS Journal of Photogrammetry & Remote Sensing*, 58:129–151, 2004.

[7] P. Barnard, J. May, D. Duke, and D. Duce. Systems, interactions, and macrotheory. *ACM Transaction on Computer-Human Interaction*, 7(2):222–262, 2000.

[8] M. Barzohar and D. Cooper. Automatic finding of main roads in aerial images by using geometric-stochastic models and estimation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 18(7):707–721, 1996.

[9] A. Baumgartner, S. Hinz, and C. Wiedemann. Efficient methods and interfaces for road tracking. *International Archives of Photogrammetry and Remote Sensing*, 34(3B):28–31, 2002.

[10] A. Baumgartner, C. Steger, H. Mayer, and W. Eckstein. Multi-resolution, semantic objects, and context for road extraction. In W. Forstner and L. Plumer, editors, *Semantic Modeling for the Acquisition of Topographic Information from Images and Maps*. Birkhäuser Verlag, 1997.

[11] A. Baumgartner, C. Steger, C. Wiedemann, H. Mayer, E. Eckstein, and H. Ebner. Update of roads in GIS from aerial imagery: verification and multi-resolution extraction. In *International Archives of Photogrammetry and Remote Sensing*, volume 31, pages 53–58, 1996.

[12] L. Bentabet, S. Jodouin, D. Ziou, and J. Vaillancourt. Road vectors update using SAR imagery: a Snake-based method. *IEEE Transaction on Geoscience and Remote Sensing*, 41(8):1785–1803, 2003.

[13] A. Blake and M. Isard. *Active Contours*. Springer, first edition, 1998.

[14] A. Blum. On-line algorithms in machine learning. In A. Fiat and G. Woeginger, editors, *Online algorithms - the state of the art*, pages 306–325. Springer, 1998.

[15] M. Brady. Computational approaches to image understanding. *ACM Computing Surveys*, 14(1):3–71, 1982.

[16] R. Brown and P. Hwang. *Introduction to random signals and applied Kalman filtering*. Wiley, second edition, 1992.

[17] T. Caelli, A. McCabe, and G. Briscoe. Shape tracking and production using hidden Markov models. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(1):197–221, 2001.

[18] J. Canny. A computational approach to edge detection. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[19] M. Carreira, M. Mirmehdi, B. Thomas, and J. Haddon. Grouping of directional features using an extended Hough transform. In *Proceedings of the 15th International Conference on Pattern Recognition*, volume 3, pages 990 – 993, Barcelona, Spain, September 2000.

[20] L. Cheng, D. Schuurmans, S. Wang, T. Caelli, and S. Vishwanathan. Online learning with sparse kernels. Technical report, University of Alberta, 2006.

[21] R. Cole, S. Vuuren, B. Pellom, K. Hacioglu, J. Ma, J. Movellan, S. Schwartz, D. Wade-Stein, W. Ward, and J. Yan. Perceptive animated interfaces: first steps towards a new paradigm for human-computer interaction. *Proceedings of the IEEE*, 91(9):1391–1405, 2003.

[22] C. Colombo, A. Bimbo, and A. Valli. Visual capture and understanding of hand pointing actions in a 3-d environment. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 33(4):677–686, 2003.

[23] M. Cord, M. Jordan, and J. Cocquerez. Accurate building structure recovery from high resolution aerial imagery. *Computer Vision and Image Understanding*, 82(2):138–178, 2001.

[24] D. Crevier and R. Lepage. Knowledge-based image understanding systems: a survey. *Computer Vision and Image Understanding*, 67(2):161–185, 1997.

[25] M. Datcu and K. Seidel. Human-centered concepts for exploration and understanding of earth observation images. *IEEE Transaction on Geoscience and Remote Sensing*, 43(3):601–609, 2005.

[26] J. Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *Journal of the Optical Society of America*, 2(7):1160–1169, 1985.

[27] R. Duda and P. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[28] Z. Duric, W. Gray, R. Heishman, F. Li, A. Rosenfeld, M. Schoelles, C. Schunn, and H. Wechsler. Integrating perceptual and cognitive modeling for adaptive and intelligent human-computer interaction. *Proceedings of the IEEE*, 90(7):1271–1288, 2002.

[29] J. Elder and R. Goldberg. Image editing in the contour domain. In *Proceedings of the IEEE Computer Vision and Pattern Recognition*, pages 374–381, Santa Barbara, CA, June 1998.

[30] M. Encarnacao and S. Stoev. An application independent intelligent user support system exploiting action-sequence based user modeling. In *Proceedings of the Seventh International Conference on User Modeling*, pages 245–254, Banff, Canada, June 1999.

[31] M. Everingham, B. Thomas, and T. Troscianko. Wearable mobility aid for low vision using scene classification in a markov random field model framework. *International Journal of Human-Computer Interaction*, 15(2):231–244, 2003.

[32] J. Fails and D. Olsen. Interactive machine learning. In *Proceedings of the 8th International Conference on Intelligent User Interface*, pages 39–45, Miami, FL, 2003.

[33] The Association for Computing Machinery. ACM SIGCHI curricula for human-computer interaction. 2006.

[34] M. Fortier, D. Ziou, C. Armenakis, and S. Wang. Survey of work on road extraction in aerial and satellite images. Technical report, Université de Sherbrooke, 2000.

[35] M. Fradkin, H. Maitre, and M. Roux. Building detection from multiple aerial images in dense urban areas. *Computer Vision and Image Understanding*, 82:181–207, 2001.

[36] A. Francois and G. Medioni. A human-assisted system to build 3-D models from a single image. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, volume I-1, pages 9282–9288, Florence, Italy, June 1999.

[37] D. Freedman and T. Zhang. Interactive graph cut based segmentation with shape priors. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, pages 755–762, San Diego, CA, June 2005.

[38] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1), 1997.

[39] Y. Furukawa and Y. Shinagawa. Accurate and robust line segment extraction by analyzing distribution around peaks in Hough space. *Computer Vision and Image Understanding*, 92(1):1–25, 2003.

[40] D. Geman and B. Jedynak. An active testing model for tracking roads in satellite images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(1):1–14, 1996.

[41] J. Geusebroek, A. Smeulders, and J. Weijer. Fast anisotropic Gauss filtering. *IEEE Transactions on Image Processing*, 12(8):938–943, 2003.

[42] I. Giakoumis, N. Nikolaidis, and I. Pitas. Digital image processing techniques for the detection and removal of cracks in digitized paintings. *IEEE Transactions on Image Processing*, 15(1):178–188, 2006.

[43] W. Glatz. RADIUS and the NEL. In O. Firschein and T. Strat, editors, *RADIUS: Image Understanding for Imagery Intelligence*. Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1997.

[44] F. Gougeon. Toward semi-automatic forest inventories using individual tree crown (ITC) recognition. *Forest Research Applications*, 22:1–6, 2000.

[45] C. Groat. The National Map - a continuing, critical need for the nation. *Photogrammetric Engineering & Remote Sensing*, 69(10), 2003.

[46] A. Gruen and H. Li. Semi-automatic linear feature extraction by dynamic programming and LSB-snakes. *Photogrammetric Engineering & Remote Sensing*, 63(8), 1997.

[47] E. Gulch. Image analysis in semi-automatic building extraction. *KI*, 4:28–31, 2001.

[48] A. Harders and G. Székely. Enhancing human-computer interaction in medical segmentation. *Proceedings of the IEEE*, 91(9):1430–1442, 2003.

[49] P. Harker and K. Lightfoot. Updating the map of Canada. *Geomatica*, 52(1):60–52, 1998.

[50] W. Harvey, J. McGlone, D. McKeown, and J. Irvine. User-centric evaluation of semi-automated road network extraction. *Photogrammetric Engineering & Remote Sensing*, 70(12):1353–1364, 2004.

[51] C. Healey, K. Booth, and J. Enns. User-centric evaluation of semi-automated road network extraction. *ACM Transaction on Computer-Human Interaction*, 3(2):107–135, 1996.

[52] C. Heipke, H. Mayer, C. Wiedemann, and O. Jamet. Evaluation of automatic road extraction. *International Archives of Photogrammetry and Remote Sensing*, XXXII:47–56, 1997.

[53] C. Heipke, C. Steger, and R. Multhammer. A hierarchical approach to automatic road extraction from aerial imagery. In D. McKeown and I. Dowman, editors, *Integrating Photogrammetric Techniques with Scene Analysis and Machine Vision II*. 1995.

[54] V. Helsinki. Digital orthophotos and map revision in national land survey of Finland. *The International Archives of Photogrammetry and Remote Sensing*, 32(4):463–466, 1998.

[55] K. Herndon and T. Meyer. 3d widgets for exploratory scientific visualization. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 69–70, 1994.

[56] S. Heuel and R. Nevatia. Including interaction in an automated modelling system. In *Proceedings of the International Symposium on Computer Vision*, pages 383–388, Coral Gables, FL, November 1995.

[57] K. Hinckley. *Haptic Issues for Virtual Manipulation*. PhD thesis, University of Virginia, USA.

[58] S. Hinz and A. Baumgartner. Automatic extraction of urban road networks from multi-view aerial imagery. *ISPRS Journal of Photogrammetry and Remote Sensing*, 58:83–98, 2003.

[59] L. Hong, Y. Wan, and A. Jain. Fingerprint image enhancement: algorithm and performance evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):777–789, 1998.

[60] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse. The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 256–265, Madison, WI, July 1998.

[61] E. Horvitz, C. Kadie, T. Paek, and D. Hovel. Models of attention in computing and communication: From principles to applications. *Communications of the ACM*, 46(3):52–59, 2003.

[62] Y. Hsieh. Sitecity: A semi-automated site modeling system. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 499–506, San Francisco, CA, June 1996.

[63] X. Hu, Z. Zhang, and C. Tao. A robust method for semi-automatic extraction of road centerlines using a piecewise parabolic model and least square template matching. *Photogrammetry Engineering & Remote Sensing*, 70(12):1393–1398, 2004.

[64] J. Hug. *Semi-automatic segmentation of medical imagery.* PhD thesis, Swiss Federal Institute of Technology, Zurich.

[65] A. Huguet, R. Carceroni, and A. Araujo. Towards automatic 3D reconstruction of urban scenes from low-altitude aerial images. In *Proceedings of the 12th International Conference on Image Analysis and Processing*, pages 254–259, Mantova, Italy, September 2003.

[66] J. Illingworth and J. Kittler. A survey of the Hough transform. *Computer Vision, Graphics, and Image Processing*, 44(1):87–116, 1988.

[67] M. Isard and A. Blake. CONDENSATION-conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[68] A. Jameson. Systems that adapt to their users. In *17th International Joint Conference on Artificial Intelligence Tutorial*, Seattle, WA, August 2001.

[69] B. Jeon, J. Jang, and K. Hong. Road detection in spaceborne SAR images using a genetic algorithm. *IEEE Transaction on Geoscience and Remote Sensing*, 40(1):22–29, 2002.

[70] R. Kalman. A new approach to linear filtering and prediction problems. *ASME Journal of Basic Engineering*, 82(D):35–45, 1960.

[71] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[72] A. Katartzis, H. Sahli, V. Pizurica, and J. Cornelis. A model-based approach to the automatic extraction of linear feature from airborne images. *IEEE Transaction on Geoscience and Remote Sensing*, 39(9):2073–2079, 2001.

[73] T. Kim, S. Park, M. Kim, S. Jeong, and K. Kim. Tracking road centerlines from high resolution remote sensing images by least squares correlation matching. *Photogrammetric Engineering and Remote Sensing*, 70(12):1417–1422, 2004.

[74] W. Kim. Computer vision assisted virtual reality calibration. *IEEE Transaction on Geoscience and Remote Sensing*, 39(9):2073–2079, 2001.

[75] J. Kivinen and M. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.

[76] D. Klang. Automatic detection of changes in road databases using satellite imagery. *The International Archives of Photogrammetry and Remote Sensing*, 32(4):293–298, 1998.

[77] H. Koike, Y. Sato, and Y. Kobayashi. Integrating paper and digital information on enhanceddesk: a method for realtime finger tracking on an augmented desk system. *ACM Transaction on Computer-Human Interaction*, 8(4), 2001.

[78] J. Kolter and M. Maloof. Dynamic weighted majority: a new ensemble method for tracking concept drift. In *Proceedings of the 3rd International Conference on Data Mining*, pages 123–130, Los Alamitos, CA, August 2003.

[79] D. LaBerge. Computational and anatomical models of selective attention in object identification. In M. Gazzaniga, editor, *The Cognitive Neurosciences*. Cambridge, MA: Bradford, 1995.

[80] I. Laptev, H. Mayer, T. Lindeberg, W. Eckstein, C. Steger, and A. Baumgartner. Automatic extraction of roads from aerial images based on scale-space and snakes. *Machine Vision and Applications*, 12(1):23–31, 2000.

[81] V. Leavers. Survey: Which Hough transform? *Computer Vision, Graphics, and Image Processing*, 58(2):250–264, 1993.

[82] M. Lee, I. Cohen, and S. Jung. Particle filter with analytical inference for human body tracking. In *Proceedings of the IEEE Workshop on Motion and Video Computing*, pages 159–166, Orlando, Florida, December 2002.

[83] S. Lee, S. Jung, and R. Nevatia. Integrating ground and aerial views for urban site modeling. In *Proceedings of the 16th International Conference on Pattern Recognition*, volume 4, pages 107–112, Quebec City, Canada, August 2002.

[84] V. Lepetit and M. Berger. A semi-automatic method for resolving occlusion in augmented reality. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 225–230, Hilton Head Island, SC, June 2001.

[85] T. Lillesand, R. Kiefer, and J. Chipman. *Remote Sensing and Image Interpretation*. John Wiley & Sons, Inc., New York, 5th ed. edition, 2004.

[86] N. Littlestone and M. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

[87] V. Madhok and D. Landgrebe. A process model for remote sensing data analysis. *IEEE Transaction on Geoscience and Remote Sensing*, 40(3):680–686, 2002.

[88] M. Maloof, P. Langley, T. Binford, R. Nevatia, and S. Sage. Improved rooftop detection in aerial images with machine learning. *Machine Learning*, 53:157–191, 2003.

[89] M. Markou and S. Singh. Novelty detection: a review - part 2: neural network based approaches. *Signal Processing*, 83(12):2499–2521, 2003.

[90] H. Mayer. Automatic object extraction from aerial imagery-A survey focusing on buildings. *Computer Vision and Image Understanding*, 74:138–149, 1999.

[91] H. Mayer and C. Steger. Scale-space events and their link to abstraction for road extraction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 53:62–75, 1998.

[92] D. McKeown, G. Bullwinkle, S. Cochran, W. Harvey, C. McGlone, J. McMahill, M. Polis, and J. Shufelt. Research in image understanding and automated cartography: 1997-1998. Technical report, School of Computer Science, Carnegie Mellon University, 1998.

[93] D. McKeown and J. Denlinger. Cooperative methods for road tracking in aerial imagery. In *Proceedings of the IEEE Conference in Computer Vision and Pattern Recognition*, pages 662–672, Ann Arbor, MI, June 1988.

[94] T. Mitchell. *Machine Learning*. McGrow-Hill Companies Inc., 1997.

[95] T. Moeslund and E. Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81(3):231–268, 2001.

[96] L. Moore. The USGS Geological Survey's revision program for 7.5-minute topographic maps. 2000.

[97] E. Mortensen. Vision-assisted image editing. *ACM SIGGRAPH*, 33(4):55–57, 1999.

[98] Y. Motai and A. Kak. An interactive framework for acquiring vision models of 3-D objects from 2-D images. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 34(1):566–578, 2004.

[99] P. Muneesawang and L. Guan. An interactive approach for CBIR using a network of radial basis functions. *IEEE Transactions on Multimedia*, 6(5):703–715, 2004.

[100] A. Murata. Improvement of pointing time by predicting targets in pointing with a PC mouse. *International Journal of Human-Computer Interaction*, 10(1):23–32, 1998.

[101] B. Myers, S. Hudson, and R. Pausch. Past, present, and future of user interface software tools. *ACM Transactions on Computer-Human Interaction*, 7(1):3–28, 2000.

[102] V. Navalpakkam and L. Itti. A goal oriented attention guidance model. In *Proceedings of the Second Workshop on Biologically Motivated Computer Vision*, pages 453–461, Tuebingen, Germany, November 2002.

[103] R. Nevatia and K. Price. Automatic and interactive modeling of buildings in urban environments from aerial images. In *Proceedings of the Ninth International Conference On Image Processing*, pages 525–528, Rochester, NY, September 2002.

[104] J. Nielsen. *Usability Engineering*. Morgan Kaufmann Publishers, San Francisco, CA, 1994.

[105] M. Okabe and S. Yamada. Interactive document retrieval with relational learning. In *Proceedings of the ACM symposium on Applied computing*, pages 27–31, Las Vegas, NV, March 2001.

[106] K. Okada, U. Akdemir, and A. Krishnan. Blob segmentation using joint space-intensity likelihood ratio test: application to 3D tumor segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 437–444, San Diego, CA, June 2005.

[107] S. Olabarriaga and A. Smeulders. Interaction in the segmentation of medical images: A survey. *Medical Image Analysis*, 5(2):127–142, 2001.

[108] M. Pantic and L. Rothkrantz. Toward an affect-sensitive multimodal human-computer interaction. *Proceedings of the IEEE*, 91(9):1370–1390, 2003.

[109] A. Paplinski. Directional filtering in edge detection. *IEEE Transactions on Image Processing*, 7(4):611–615, 1998.

[110] V. Pavlovic, R. Sharma, and T. Huang. Visual interpretation of hand gestures for human-computer interaction: a review. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 19(7):677–695, 1997.

[111] J. Peng, B. Bhanu, and S. Qing. Probabilistic feature relevance learning for content-based image retrieval. *Computer Vision and Image Understanding*, 75(1-2):150–164, 1999.

[112] G. Priestnall, M. Hatcher, R. Morton, S. Wallace, and R. Ley. A framework for automated extraction and classification of linear networks. *Photogrammetric Engineering and Remote Sensing*, 70(12):1373–1382, 2004.

[113] L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[114] A. Robinson, J. Morrison, P. Muehrcke, A. Kimerling, and S. Guptill. *Elements of Cartography*, chapter 1, page 4. John Wiley & Sons Inc., 1995.

[115] E. Ross. Intelligent user interfaces: survey and research directions. Technical report, Department of Computer Science, University of Bristol.

[116] Y. Rui and Y. Chen. Better proposal distributions: Object tracking using unscented particle filter. In *Proceedings of the Eighth International Conference On Computer Vision*, volume 1, pages 786–793, Kauai, Hawaii, December 2001.

[117] Y. Rui, T. Huang, and S. Chang. Image retrieval: current directions, promising techniques, and open issues. *Journal of Visual Communication and Image Representation*, 10(1):39–62, 1999.

[118] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4):288–297, 1990.

[119] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. Smola, and R. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13:1443–1471, 2001.

[120] A. Shackelford and C. Davis. A combined fuzzy pixel-based and object-based approach for classification of high-resolution multispectral data over urban areas. *IEEE Transaction on Geoscience and Remote Sensing*, 41(10):2354–2363, 2003.

[121] J. Shufelt. Performance evaluation and analysis of monocular building extraction from aerial imagery. *IEEE Transaction on Pattern Recognition and Machine Intelligence*, 21(4):311–326, 1999.

[122] M. Song and D. Civco. Road extraction using SVM and image segmentation. *Photogrammetric Engineering and Remote Sensing*, 70(12):1365–1371, 2004.

[123] B. Southall and C. Taylor. Stochastic road shape estimation. In *Proceedings of the Eighth International Conference On Computer Vision*, volume 1, pages 205–212, Vancouver, Canada, July 2001.

[124] R. Srihari and Z. Zhang. Show&tell: a semi-automated image annotation system. *IEEE Multimedia*, 7(3):61–71, 2000.

[125] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[126] J. Ton, J. Sticklen, and A. Jain. Knowledge-based segmentation of Landsat images. *IEEE Transactions on Geoscience and Remote Sensing*, 29(2):222–232, 1991.

[127] F. Tupin, B. Houshmand, and M. Datcu. Road detection in dense urban areas using SAR imagery and the usefulness of multiple views. *IEEE Transaction on Geoscience and Remote Sensing*, 40(11):2405–2414, 2002.

[128] USGS. *Standards for 1:24000-Scale Digital Line Graphs and Quadrangle Maps*. U.S. Geological Survey, U.S. Department of the Interior, 1996.

[129] USGS. *Standards for Raster Feature Separates Version 1.0*. U.S. Geological Survey, U.S. Department of the Interior, 2002.

[130] V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[131] T. Veen and F. Groen. Discretization errors in the Hough transform. *Pattern Recognition*, 14(1-6):137–145, 1981.

[132] M. Virvou and K. Kabassi. Reasoning about user's actions in a graphical user interface. *Human-Computer Interaction*, 17(4):369–398, 2002.

[133] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

[134] G. Vosselman and J. Knecht. Road tracing by profile matching and Kalman filtering. In *Proceedings of the Workshop on Automatic Extraction of Man-Made Objects from Aerial and Space Images*, pages 265–274, Birkhaeuser, Germany, April 1995.

[135] F. Wang and R. Newkirk. A knowledge-based system for highway network extraction. *IEEE Transactions on Geoscience and Remote Sensing*, 26(5):525–531, 1988.

[136] M. Ware, E. Frank, G. Holmes, M. Hall, and I. Witten. Interactive machine learning - letting users build classifiers. *International Journal of Human-Computer Studies*, 55(3):281–292, 2001.

[137] G. Welch and G. Bishop. An introduction to the Kalman filter. Technical Report TR95-041, Department of Computer Science, University of North Carolina - Chapel Hill.

[138] Christian Wiedemann and Helmut Mayer. Automatic verification of roads in digital images using profiles. In *DAGM-Symposium*, pages 609–618, 1996.

[139] D. WIlkins, S. Miller, and A. Walker. Increasing the productivity of CAD-based feature extraction through the exploitation of softcopy photogrammetry. In *ASPRS 2000 Annual Conference CD Proceedings*, Washington D.C., May 2000.

[140] D. Xiong and J. Sperling. Semiautomated matching for network database integration. *ISPRS Journal of Photogrammetry & Remote Sensing*, 59:35–46, 2004.

[141] D. Young and A. Gray. Semi-automatic boundary detection for identification of cells in dic microscope images. In *Proceedings of the 6th International Conference on Image Processing and its Applications*, pages 346–350, Dublin, Ireland, July 1997.

[142] S. Zhai, P. Milgram, and W. Buxton. The influence of muscle groups on performance of multiple degree-of-freedom input. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 308–315, Vancouver, Canada, 1996.

[143] C. Zhang. Towards an operational system for automated updating of road databases by integration of imagery and geodata. *ISPRS Journal of Photogrammetry & Remote Sensing*, 58:166–186, 2004.

[144] C. Zhang, E. Baltsavias, and A. Gruen. Knowledge-based image analysis for road reconstruction. *Asian Journal of Geoinfomatics*, 1(4):3–14, 2001.

[145] X. Zhang and H. Burkhardt. Grouping edge points into line segments by sequential Hough transformation. In *Proceedings of the 15th International Conference on Pattern Recognition*, volume 3, pages 672–675, Barcelona, Spain, September 2000.

[146] J. Zhou, W. Bischof, and T. Caelli. Understanding human-computer interactions in map revision. In *Proceedings of the 10th International Workshop on Structural and Syntactic Pattern Recognition*, pages 287–295, Lisbon, Portugal, August 2004.

[147] J. Zhou, W. Bischof, and T. Caelli. Human-computer interaction in map revision systems. In *CD-ROM Proceedings of the 11th International Conference on Human-Computer Interaction*, Las Vegas, Nevada, July 2005.

[148] J. Zhou, W. Bischof, and T. Caelli. Robust and efficient road tracking in aerial images. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (CMRT05)*, volume XXXVI, pages 35–40, Vienna, Austria, August 2005.

[149] P. Zhu, Z. Lu, X. Chen, K. Honda, and A. Eiumnoh. Extraction of city roads through shadow path reconstruction using laser data. *Photogrammetric Engineering & Remote Sensing*, 70(12):1433–1440, 2004.

[150] A. Zlotnick and P. Carnine. Finding road seeds in aerial images. *CVGIP: Image Understanding*, 57(2):243–260, 1993.

[151] J. Zou and G. Nagy. Evaluation of model-based interactive flower recognition. In *Proceedings of the 17th International Conference on pattern Recognition*, volume 2, pages 311–314, Cambridge, UK, August 2004.

Appendix A

Table of Abbreviations

The table below gives a list of all abbreviations used in this thesis and their full descriptions.

1-SVMs	One-Class Support Vector Machines
CAD	Computer-Aided Design
CCKF	Cross Correlation with Kalman Filtering
CCPF	Cross Correlation with Particle Filtering
CMU	Carnegie Mellon University
DEM	Digital Elevation Model
DOQ	Digital Orthophoto Quadrangle
EKF	Extended Kalman Filter
EM	Expectation-Maximization
GIS	Geographical Information Systems
GOMS	Goals, Operators, Methods, and Selection rules
HCI	Human-Computer Interaction
HMM	Hidden Markov Model
IML	Interactive Machine Learning
MAP	maximum *a posteriori*
MDL	Microstation Development Language
MRF	Markov Random Field
NRCan	Natural Resources Canada
NTS	National Topographic Systems
OLND	Online Learning and Novelty Detection
OS	Ordnance Survey, United Kingdom
RKHS	Reproducing Kernel Hilbert Space
RGR	Raster Graph Revision
SBSM	State Bureau of Survey and Mapping, China
SVMs	Support Vector Machines
USGS	United States Geological Survey